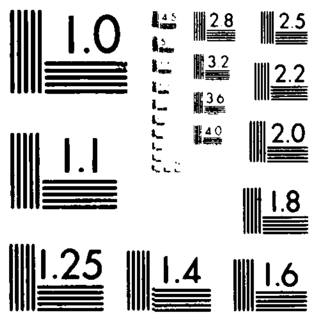
AD-A117 603 TRANSPORTATION SYSTEMS CENTER CAMBRIDGE MA F/G 1/5
AIRPORT LANDSIDE. VOLUME V. APPENDIX B. ALSIM SUBROUTINES.(U)
JUN 82 L MCCABE, M GORSTEIN
UNCLASSIFIED DOT-TSC-FAA-82-4-5 FAA-EM-80-8-5 NL

1 of 4
AD
A117603




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(11)

FAA-EM-80-8-V
DOT-TSC-FAA-82-4-V

# Airport Landside

## Volume V: Appendix B
## ALSIM Subroutines



L. McCabe
M. Gorstein

Transportation Systems Center
Cambridge MA 02142

June 1982
Final Report

This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.



U.S. Department of Transportation
**Federal Aviation Administration**

Office of Systems Engineering and Management
Washington DC 20591

Technical Report Documentation Page

| | | |
|---|---|---|
| 1. Report No. FAA-EM-80-8-V | 2. Government Accession No. A117603 | 3. Recipient's Catalog No. |

4. Title and Subtitle
AIRPORT LANDSIDE
VOLUME V: APPENDIX B ALSIM
SUBROUTINES

5. Report Date
June 1982

6. Performing Organization Code
DTS-53

7. Author(s)
L. McCabe & M. Gorstein

8. Performing Organization Report No.
DOT-TSC-FAA-82-4-V

9. Performing Organization Name and Address
U.S. Department of Transportation
Research and Special Programs Administration
Transportation Ssytems Center
Cambridge MA 02142

10. Work Unit No. (TRAIS)
FA032/R113

11. Contract or Grant No.

12. Sponsoring Agency Name and Address
U.S. Department of Transportation
Federal Aviation Administration
Office of Systems Engineering Management
Washington DC 20591

13. Type of Report and Period Covered
Final Report
Jan 1978 Sep 1980

14. Sponsoring Agency Code
ACT-420

15. Supplementary Notes

16. Abstract

This Appendix describes the operation of ten subroutines used to support the AUXILIARY and MAIN programs of ALSIM. Flow charts and listings of all programs are provided. The major portion describes the FORTRAN subprogram FORTM which is used to read input data, assign values to matrix elements, perform matrix searches and assign parameters to GPSS transactions during simulation model execution.

Six other subroutines, mostly written in IBM System/370 Assembly Language, are used in the initialization phase of the simulation to link FORTM to the MAIN program and to provide an in-core read and write capability. Two additional assembly language subroutines and a FORTRAN subroutine are used during simulation of the airport landside. The first assembly language subroutine assigns the number of passenger bags to be retrieved by the deplaning passenger transaction and generates random numbers to simulate waiting times at the bag claim facility. The second subroutine performs the same function as ASSIGN and LOGIC blocks of GPSS, but is FORTRAN callable [illegible] FORTRAN subroutine of this group detects argument errors of the pr[illegible] subroutine and prints error messages.

Other volumes of the Airport Landside report are: Volume I: Planning Guide; Volume II: The Airport Landside Simulation (ALSIM) Description and Users Guide; Volume III: ALSIM Calibration and Validation; and Volume IV: Appendix A ALSIM Auxiliary and MAIN programs

| 17. Key Words | 18. Distribution Statement |
|---|---|
| Not Applicable | DOCUMENT IS AVAILABLE TO THE U.S. PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | 328 | |

# SUMMARY

This appendix contains detailed descriptions of subroutines used during the operation of the Airport Landside Simulation Model. The major portion of this volume describes the FORTRAN subprogram LINKC, alias FORTM which is closely linked to the GPSS-V AUXILIARY or MAIN programs during program execution. FORTM expedites the flow of GPSS-V transactions within the model by performing matrix searches and assigning values to transaction parameters.

Three major program sections of FORTM are described in this document. A non-executable section consists of FORTRAN variable definition, data equivalent and namelist statements. An input section consisting of 20 subsections initializes variables, reads input data and assigns values to GPSS matrix elements. The main section of this subprogram consists of 26 subsections which assign values to the GPSS transaction parameters at each type of simulated facility. During program operation, the GPSS-V MAIN program repeatedly calls the main section of this subprogram as transactions move from one simulated facility to the next.

This document also describes a set of nine other subroutines called by the GPSS-V MAIN or AUXILIARY programs or the FORTM subprogram. A description of the purpose, usage, restrictions and program logic is included for each subroutine.

Most of the subroutines described are utilized in the initialization stage of the simulation. Subroutines CLINK, CLINK1 and CLINK2 establish linkages between the GPSS program and the FORTM subprogram, permitting HELPA blocks to operate as HELPC blocks. Subroutine MNLINK allows the simulation user to code identical mnemonics in the GPSS program and FORTM subprogram and transfer numerical values in either direction. Subroutine XCODE provides an in-core read and write capability and is used in reformatting input data read under FORTRAN format control for subsequent re-reading. Function subprogram MHBASE/MXBASE/MLBASE provides the base addresses of the GPSS-V halfword, fullword and floating point matrices used in FORTM.

The three remaining subroutines are used during the simulation phase of ALSIM. Subroutine ASSIGN/LOGIC/PVAL/FPVAL is used to assign values to the active transaction parameters, to set logic switches, or to obtain a parameter value from the active transaction. Subroutine ARGERR is called by ASSIGN/LOGIC/PVAL/FPVAL to print a message when an error in the argument list of one of these entries is detected. Subroutine BAGS assign the number of bags to be claimed by the deplaning passenger transaction and generates random numbers for subsequent use in simulating delivery times.

Several of these subroutines branch to locations or subroutines utilized by the IBM Program Product General Purpose Simulation System V -OS (5734-XS2). The documentation containing descriptions of most of the branch addresses is contained in Chapter 12 of the "General Purpose Simulation System V User's Manual" (SH20-0851). However, the subroutines providing logic set and reset capabilities in subroutine ASSIGN branch to locations internal to GPSS-V and could become obsolete if unreleased changes affecting program performance were performed by IBM. The subroutine XCODE branches to a location within IBCOM and relies on maintenance of current operational instructions and register conventions for continued successful operation.

The block diagram in Figure 1 illustrates the program levels of ALSIM. Subroutines BAGS, FORTM and CLINK are called by GPSS-V HELP, HELPA and HELPC blocks, respectively. BAGS is an IBM System/370 Assembly Language subroutine. The subroutines FORTM and CLINK are both written in FORTRAN and use CALL instructions or function references to access programs in the next lower level. With the exception of the FORTRAN subroutine MXBASE/MHBASE/MLBASE, subroutines at the third level are written in IBM System/370 Assembly Language. Branching to ARGERR from ASSIGN/LOGIC/PVAL/FPVAL is discussed in the document.

The blocks FORTM, LINKC and CLINK2 require explanation. The proper name of the FORTRAN subprogram is LINKC and contains the entry point FORTM. All calls made to this subprogram from the GPSS-V programs are HELPA calls to the entry point FORTM. LINKC

is never called explicitly by the main or auxiliary programs. When the first HELPA call is made to FORTM, this subprogram subsequently calls the assembler program CLINK2. Subprogram LINKC is then called by subroutine CLINK2. This procedure is only performed once. Control returns to CLINK2 before the entry point FORTM is reached. This operation is performed to provide linkage between FORTM and the GPSS-V programs. Details are explained in this appendix.

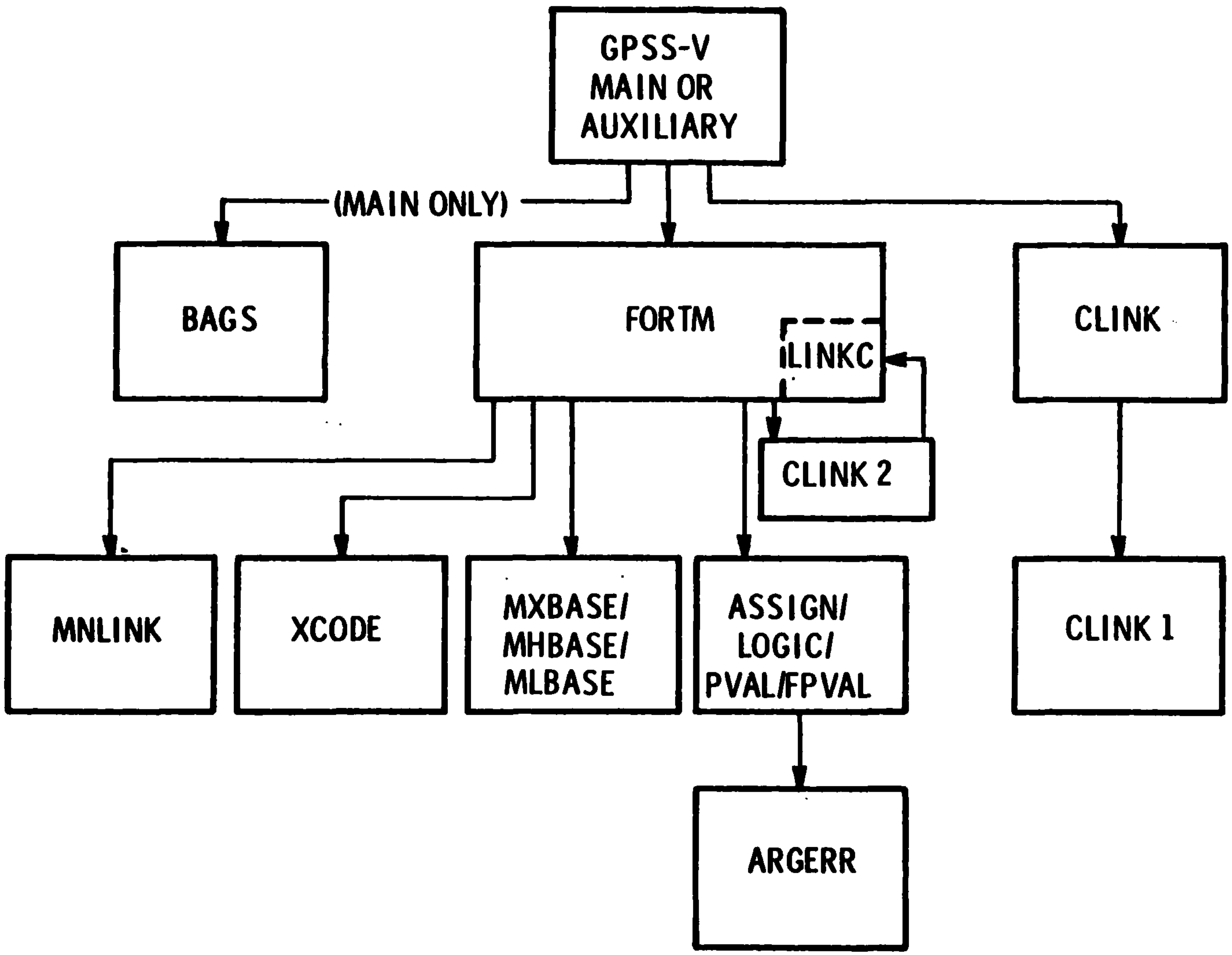



FIGURE 1. ALSIM PROGRAM LEVELS

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

# APPENDIX B-1

## LINK C (FORTM) SUBPROGRAM OF THE AIRPORT LANDSIDE SIMULATION MODEL (ALSIM)

# 1.0 INTRODUCTION

The FORTRAN portion of the Airport Landside Simulation program is called by the GPSS program to perform four major functions. These are: (1) the reading of data cards specifying airport operation; (2) filling GPSS matrices using the input data; (3) the moving of passengers from node to node by assigning transaction parameters; and, (4) the formatting of GPSS and other output statistics as summaries.

This report documents the FORTRAN program, named FORTM, and is divided in three sections. The first is the NON-EXECUTABLE STATEMENTS SECTION which contains a description of all the declarations, equivalence, namelist and data statements needed to define and initialize variables. The second is the INPUT SECTION which contains a description of how data is read into the program and the initialization process for the input and other variables. The third section is called the MAIN SECTION and contains a description of how the program handles the various calls from the GPSS program and assigns new values for parameters at each type of landside facility.

Flowcharts and a listing of the program are also included in the appendices.

## 2.0 NONEXECUTABLE STATEMENTS SECTION

This section starts with the subroutine LINKC statement which has the standard GPSS list of values to be passed and a set of INTEGER, REAL, and DIMENSION cards which set up the HELPC type link to the GPSS program. Next a list of INTEGER, REAL, DIMENSION, and DATA statements define and initialize numerous variables. A data statement then places the names of all the facility types in the array FACTYPE. The order in which the enplaning curb areas are searched for a vacant space is placed by a data statement in the array IEPSCH. A final data statement then places the full title of the facilities as written on input data cards in array NAMER8.

Equivalence and namelist statements are described in Tables 1 and 2 respectively. A set of statement functions follow which use bases, addresses, numbers of columns and row-column identifier of each element to compute the locations of the GPSS matrix elements. This section ends with a RETURN.

TABLE 1. EQUIVALENCE STATEMENTS

| ARRAY OR SCALAR NAME | EQUIVALENCED TO |
|---|---|
| DUM8(1)<br>IDUM1(1 to 22) | Input values to be zeroed before new input line is read in. |
| NFASCM (1 to 15, 1)<br><br>NFASCM (1 to 20, 2) | Names of scalars identifying numbers of facility types.<br>INDEXF (1 to 20) Index number of facility type. Add facility number in type for MH9 row. |
| FACQSX (1 to 14) | Scalars which contain the base values assigned each facility type by the GPSS compiler. |
| NSORT (Integer*4) | NSORTD (1 to 2) (Integer*2)<br>Allows the section of the program that sorts the facilities to sort MH9 by facility number and by facility type in a single pass. |
| FROMTO (1 to 2) | FROM, TO |

TABLE 2. NAMELIST STATEMENTS

| NAMELIST NAME | USAGE | DEFAULTS |
|---|---|---|
| AL | Airline cards | |
| BU | Bus/Limousine Card | ARVBUS = 0<br>DEPBUS = 0 |
| FL | Arriving and departing flight cards | DOM = 1<br>AIRLIN = DEFLIN<br>TPAX = 0 |
| GE | Facility location cards | |
| GT | Ground Transportation cards | |
| OV | Walking Time Override cards | |
| PA | Parameter card | WWGATE =0.0 LEAVEL =15<br>GRGATE =0.0 LEAVEC =10<br>BOARDT =15 min LEAVEV =10<br>ERRORS =50 |
| S | Storage cards | |
| ST | Initial cards | SCALE = 1<br>DSTFAC = 1.1<br>WALKSP = 1.0 meter/sec. |
| TI | % Preticketed card | |
| TR | Transfer Flight card | ADD = 7200 sec.<br>DELETE = 1800 sec. |
| CH | Server Change | |
| TS | Time Series Output | |

# 3.0 INPUT SECTION

## 3.1 INITIAL SECTION

The first statement in this section is an ENTRY statement with the six element array IVALUE passed as a parameter. The program then branches to the statement number which has the same value as IVALUE(1).

If IVALUE(1) is 1, the program goes to statement number 1, which is the start of the input section. Variables used for counters are set to zero and default values are set to those listed in Table 2, with the exception of those under namelist FL.

The first input card is then read. If the card is the JOBTAPE card, a flag is set to indicate that the GPSS auxiliary program is being used, and the next card is then read. If this card is a comment card, indicated by an asterisk in the first column, the next card is read. This card, which should be the INITIAL card, is written to main memory and read with a namelist format of ST. The simulation start and finish times, default bag claim area DEFBAG, the default airline DEFLIN, a factor DSTFAC accounting for non-direct paths between points, a scale factor, and a walking speed are contained on this card. Subroutine MNLINK is then called to set up the mnemonic link transfer from either of two calling statements,

depending upon whether the auxiliary or the main program is using the program. Subroutine CLINK2 is then called to transfer the address list from GPSS. For the main program, the contents of the variables containing the default values for the time of adding, ADD, or deleting, DELETE, from the transfer flight table in seconds are placed in their respective savevalues, XFADH and XFDXH. A scaling factor, SCLXH, is used to allow GPSS transactions to represent N passenger groups. Starting locations of GPSS matrices are computed using the functions MHBASE and MLBASE. The contents of the variables containing the times for the start, START, in hours, and end, FINISH, in seconds, of the simulation are placed in their respective savevalues, CLKXH and ENDXF.

The section of the program that is used to read in the rest of the cards then follows. The area of main memory that will contain the input values is zeroed out first. The variable TWOWAY is blanked out. A card is then read in, and the counter, NCARD, for the number of cards read in and the counter for the number of output lines, LINECT, are incremented. If the counter, LINECT, for the number of output lines exceeds 50, then the counter is set back to one and the page title 'INPUT DATA' is printed at the top of the next page. The line is then printed out with a line number. If the card is a comment card then the program branches back to the section that reads in the next card. If the card is not a comment card, the program next branches to the section that handles the type of the input card. For the geometry input cards the card

identifier is compared with the array FACTYP. When the matching facility is found the program notes the facility type number, I, and then branches to the geometry input section. If the card is not a recognized input type the program prints out an error message; sets an error flag, NERRSW; assigns 1000 to PH1; and branches back to the section that reads the next card in.

## 3.2 FLIGHT SCHEDULE INPUT

The input line is written into main memory and then read again with a namelist format of FL. The counter, NROW, for the number of rows in the Flight Schedule Matrix Savevalue, MH1, is incremented by one. Next the value of the GATE, PAX, and TIME variables is checked. If any of these variables have a value of zero then the program prints an error message; sets an error flag, NERRSW; assigns 1000 to PH1; and branches back to the section that reads the next card in. Next, the program tests whether the flight is an arrival or departure flight. If the flight is a departure then the program determines if both the default airline and the input AIRLIN are zero. If both are zero the program proceeds as in the previous error condition. Otherwise the program sets MH1(NROW,1) to 1, to indicate a departure flight. Next MH1(NROW,2) is set equal to the input flight number, FLTNO. The program then determines if the variable AIRLINE has been specified in the input. If it has

not, the AIRLIN is set equal to the default airline, DEFLIN. Then MH1(NROW,3) is set equal to AIRLIN. MH1(NROW,4) is set equal to TIME, the scheduled arrival or departure time. MH1 (NROW,6) is then set to time of flight from start in minutes. Next, MH1(NROW,7) is set to 1, 2, or 3 for DOMESTIC, COMMUTER or INTERNATIONAL flights respectively. MH1(NROW,8) is set to aircraft type, AC. MH1(NROW,9) is next set to the gate number, GATE. If the input BAG is zero and if it is an arrival flight, then BAG is set equal to the default baggage area number, DEFBAG. If BAG is still zero and it is an arrival flight, then the program prints an error message; sets an error flag, NRRSW; assigns 1000 to PH1; and branches back to read in the next card. If BAG is non-zero, MH1(NROW,12) is then set equal to BAG, the baggage area number. If the SCALE is not equal to 1, then MH1(NROW,10) is set equal to PAX, the number of terminating or originating passengers on the flight, divided by SCALE plus 0.51 to round to a whole integer; and MH1 (NROW,11) is set equal to TPAX(1), the number of transfer passengers in the flight, divided by SCALE plus 0.51. If the scale is equal to 1 then MH1(NROW,10) and MH1(NROW,11) are set equal to PAX and TPAX(1) respectively.

For simulations of a single concourse, with transfer passengers originating on other concourses, the input value TPAX(2) is placed in MH1(NROW,13). If transit passengers are simulated, TPAX (3) is placed in MH1(NROW,16). These two quantities are scaled as PAX and TPAX (3). The program then branches back to the section that reads in the next card.

## 3.3 TIME SERIES SPECIFICATIONS

The program writes the input line to main memory and reads the record with namelist name TS. Values of GPSTO, GPQUE or GPHALF elements are read into their respective array. These values are the absolute numbers of the GPSS storages, queues, or halfword savevalues selected for time series printouts. Flow and queue length values are produced periodically during this simulation run for the specified GPSS storages and queues. GPSS halfword savevalues are also output and are used to represent flow at specified GPSS program areas.

## 3.4 AIRLINE DATA INPUT

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the jobtape flag is not set, the input line is then written into main memory and read with a namelist format of AL. For each airline, J, specified, MH2(J,1) is set equal to EPCURB, the enplaning curb number; MH(J,2) is set equal to the percent of preticketed passengers using express check-in times 10, EXPCHK*10; and MH2 (J,3) is set equal to BUSTOP, the bus stop area number for enplaning passengers. The program next branches back to the section that reads in the next card.

## 3.5 GROUND TRANSPORTATION INPUT

Input variables are first initialized to zero. The program writes the input line to main memory and reads with a namelist format of GT. All variables read in are divided by 100 to obtain percentages. The variable I is set equal to 1, 2, or 3

for DOMESTIC, COMMUTER or INTERNATIONAL flights respectively. If the jobtape flag is set, the program places the cumulative percentages for private car, rented car, bus and taxi respectively for the auxiliary program in ML2(I,1 through 4). If the jobtape flag is not set, the program places the cumulative percentages for, rental, bus, and taxi respectively with private car excluded in ML2(I,2 through 4). The program then branches back to read in the next card.

3.6 %PRETICKETED PASSENGER INPUT

The program writes the input line to main memory and then reads with a namelist format of TI. The program then places in MH4(1 through 3, 1) the percent of preticketed passengers*10 for DOMESTIC, COMMUTER, and INTERNATIONAL flights respectively. Next, the program places in MH4(1 through 3, 2) the percent of preticketed direct *100 divided by % preticketed if both the % preticketed variable and the % preticketed direct variables are greater than 0. The program then branches back to the section that reads in the next card.

3.7 WALKING TIME/DISTANCE OVERRIDE INPUT

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the jobtape flag is not set, the program writes the input line to main memory and reads it with a namelist format of OV. If the input walking time, TIME, is equal to zero, which indicates that the distance, DIST, was input instead, TIME is set equal to DIST/WALKSP, the walking distance divided by the walking

speed. The program then places the walking time, TIME in MH6 (FROM, TO) and MH6(TO, FROM). The program then branches back to the section that reads in the next card.

3.8 PARAMETER CARD INPUT

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the jobtape flag is not set, the program writes the input line to the main memory and reads in the variables with a namelist format of PA. The program then branches back to the section that reads in the next card.

3.9 BUS SCHEDULE INPUT

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the jobtape card is not set, the program writes the input to main memory and reads with a namelist format of BU. The program then places in savevalue ABUXH, ARBUS*60, the interval in seconds between bus arrivals. Next, the program places in savevalue DBUXH, DEPBUS*60, the interval in seconds between bus departures. The program then branches back to the section that reads in the next card.

3.10 GPSS STORAGE CAPACITY

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the job tape flag is not set, the program writes the input to main memory and reads with a namelist format of S. For each storage specified on the input card the number of available units for that storage

is set equal to the input specified. The program then branches back to the section that reads in the next card.

### 3.11 TRANSFER FLIGHT OVERRIDE INPUT

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the jobtape card is not set, the program writes the input line to main memory and reads with a namelist format of TR. If the input variable ADD is greater than zero, then the time for adding a flight to the transfer flight table in seconds, ADD*60, is placed in savevalue XFAXH. If the input variable, DELETE, is greater than zero, then the time for deleting a flight from the transfer flight table in seconds, DELETE*60, is placed in savevalue XFDXH. The program then branches back to the section that reads in the next card.

### 3.12 RUN TITLE CARD INPUT

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the jobtape flag is not set, the program determines if there are more than 5 title lines. If there are, an error message will be written stating that only 5 title lines can be input and that the current line will not be used, and then the program branches back to the section that reads in the next card. If the number of title lines does not exceed 5, then the program increments the counter, NTLINS, for the number of title lines by one. Next, the input line is written to main memory and read into array ITITLE. The program then branches back to the section that reads in the next card.

3.13 GEOMETRY INPUT

If the jobtape flag is set, the program branches back to the section that reads in the next card. If the jobtape flag is not set, the element of FACOSX corresponding to the facility type number, I, is obtained. This element is the GPSS identifier number for the first queue-storage entity of this type. Two (2) is then subtracted from this number to aid in accessing the Nth facility of this type. This is performed for two reasons, each requiring a subtraction by unity.

This value is first decremented by one so that the Nth facility of a class may be directly referenced if the value of N is one or greater. If M represents the number of the first facility of the Ith class, the Nth facility is identified as the M+N-1 landside facility. One is subtracted from M for convenient reference. For example, if the gates have been assigned storage numbers 25 through 42 in the GPSS program and the variable GAQSL or M, representing the first gate facility, is also defined as 25, subtracting one from this value allows the referencing of the Nth entity of this type, where, in this example N ranges from 1 to 18. Thus 24 + N identifies the GPSS storage number for the Nth gate.

The second value of one is subtracted because of the nature of addressing GPSS arrays containing entity information. One objective of the facility data card is to provide the GPSS program with the number of available service units at a particular facility. This is performed by placing the number of servers from input data into the standard array ISTO. The

location of the element is computed in FORTRAN. When the Nth member of a specific entity type is addressed, the formula for locating the subscript of the ISTO matrix contains an N-1 term when referring to the Nth entity index number. To continue the above example, the subscript K, of ISTO, when used in reference to the Nth gate, is given by M+N-2 or 23+N.

Following the location of the ISTO MATRIX, the program sets the variable NOFAC to the value I, the number of the facility type. The program then blanks out long facility name titles if necessary. Next, the input line is written to main memory and read with a name list format of GE. If the error flag, NERRSW, is set, the program branches back to the section that reads in the next card. If the input value of the X or Y coordinate is not equal to zero, the value are placed in MH3(I, 1 to 2) respectively, where I is the point number. If the exit point, EXITPT, or entrance point, ENTRPT, are specified as other than the nearest one to the Ith point, they are entered in MH3(I,3) and MH3 (I,4); otherwise the program will later compute these.

The program then processes from one to four facilities of one type which are allowed on one input line. For each facility specified on the input line, the counter for the total number of facilities NGEO, is incremented by 1, and the counter for the number of facilities of a given type, NFASCM(NOFAC,1), is also incremented by 1. For each facility, MH9(NGEO,1) is set equal to the facility type NOFAC; MH9(NGEO,2) is set equal to the facility number in type, FACNO(I); and MH9(NGEO,3) is set equal to the point number, POINT, respectively. If the point number

of the facility being processed is greater than the previous maximum point number, MAXPT, then the maximum point number is set equal to the current point number. If a size for the facility is nonzero, SIXE (I), the number of available units in storage for that facility is set equal to ISTO(k). For a zero value of SIZE (I), the program assigns the GPSS default value for storage size.

When enplaning and deplaning curbside facilities are being processed, sizes of each are divided by the scale factor and ISTO(k) is redefined by the result. For each of these facility types, storages are designated in the GPSS program for double parking and queuing. The sizes of each storage are specified by input variable DPARK and CURBQ, respectively. When an enplaning or deplaning curbside data card is processed, the double parking and queue storage numbers, K, are calculated and ISTO(k) is made equal to DPARK or CURBQ. A default value of one is used if either size is zero.

Parameters specific to each facility type are equivalenced to elements of the array IPARAM. These are placed in columns 4 through 6 of MH9.

Terminal entrance and exits are assumed to be bi-directional unless the parameter, TWOWAY = NO, is specified on the data card. If the facility type is not an entrance or an exit, the program branches back to the section that reads in the next input card. When the facility type is an entrance or exit, then the program determines if the variable TWOWAY is set equal to 'NO'.

TWOWAY can be set to 'NO' by the input line, which means that the entrance or exit specified is only an entrance or an exit, or TWOWAY can be set to 'NO' by the program to indicate that the program has already created a side-by-side entrance/exit for this facility. If TWOWAY is 'NO' then the program branches back to the section that reads in the next input line. For TWOWAY not equal to 'NO', and an exit card input, the variable for the facility type, I, is set equal to 6, the number for an entrance. If TWOWAY is 'NO1; and an entrance card is input, the variable I is set equal to 7, the number for an exit. The program sets TWOWAY equal to 'NO' and branches back to the start of the Geometry Input Section to define the other side of the entrance/exit pair.

### 3.14 FLIGHT SCHEDULE SORTING SECTION

The program branches to this section when the end of file is encountered when reading in the input line. If the error flag, NERRSW, has been set then the program branches to statement number 99999 which is a RETURN. The flight schedule in MH1 is sorted by time after simulation start in column 6. The value -1, is then placed in MH1(NROW+1,1) to indicate the end of the flight schedule. If the jobtape flag is set, the program writes the message 'END OF INPUT DATA' and branches to statement number 99999 which is a RETURN. If the jobtape flag is not set then the program goes to the FACILTIY SORT SECTION.

3.15 FACILITY SORT SECTION

The flag, NSWTC1, is placed in a reset condition, then the program sorts the facility table, MH9, by facility type and number in type. Facility type and number in type are sorted in one pass because the type and number for each entity are placed in one word, NSORT. Following this sort, NSTCW1 is tested. If it is set, then the program branches to the SET UP FACILITY POINTER TABLE SECTION. If the flag, NSWTC1, is in a reset condition the program determines if any numbers have been skipped or if a duplication of facility numbers exists in the defining of facilities in MH9. If there have been skipped facility numbers the program creates dummy facilities in MH9 using the numbers that have been skipped. Doubly defined facilities terminate the simulation. The program sets the flag, NSWTC1, and branches back to again sort MH9 and performs the subsequent test on NSTWC1. If there are no skipped facility numbers in MH9, the program then goes to the SET UP FACILITY POINTER TABLE SECTION.

3.16 SETUP FACILITY POINTER TABLE SECTION

This section sets up the facility pointer table, MH8, which is the same as the array NFASCM. The program places in MH8 (1 through 20,1) the number of the facility in its type, from NFASCM(I,1). The program then places in MH8 (1 to 20,2) the index number of the facility, NFASCM(I,2), which is the number of rows in MH9 before this facility type. The program then goes on to the POINT-TO-POINT WALKING TIME CALCULATING SECTION.

3.17 POINT-TO-POINT WALKING TIME CALCULATION SECTION

The program calculates the walking time for each pair of points and stores it in MH6. If both the X and Y coordinates are zero for a point, indicating a possibly undefined point, then a message is written indicating the point with (0,0) coordinates. If the walking time for a point-to-point pair was previously input in the WALKING TIME/DISTANCE OVERRIDE INPUT SECTION then the value for that point-to-point pair is left as defined. The program then goes on to the DETERMINE CLOSEST ENTRANCE AND EXIT TO EACH POINT SECTION.

3.18 DETERMINE CLOSEST ENTRANCE AND EXIT TO EACH POINT SECTION

The program determines the closest entrance and exit to each point and stores it in MH3 (1 to MAXPT, 3 to 4) respectively. If the closest entrance or exit was previously defined in the GEOMETRY INPUT SECTION then the value for that entrance or exit is used. The program then goes on to the CHECK FOR UNDEFINED FACILITY SECTION.

3.19 CHECK FOR UNDEFINED FACILITY SECTION

The program checks the array, NFASCM, to determine if any facilities have not been defined. For undefined facilities the program writes a message which includes the statement that the following facilities are undefined, the list of the undefined facilities, and the statement that execution continues. The program then goes on to the END OF INPUT SECTION.

3.20 PARAMETER ASSIGNMENT AND END OF INPUT SECTION

The program sets the savevalue, BDTXE, equal to the

boarding time, BOARDT, in seconds. The latest times of transit and transfer passengers for leaving lobby and concessions, LEAVEL and LEAVEC, and the spread, LEAVEV, of the uniform random distribution modifying these times are converted from minutes to seconds. Percentages of well-wishers proceeding to the gate, vehicles proceeding from enplaning curbside to parking and percentages of enplaning ticketed passengers using curbside check-in are multiplied by 10 and converted to savevalues, WWGXH, CPKXH, and CRBXH respectively.

The percentage of terminating passengers with greeters, GREET, is divided by 100 and placed in the floating point savevalue GRTXL. The percentage of greeted terminating passengers greeted at curbside, CRBGT, is divided by 100 and placed in the floating point savevalue CGTXL. The percentage of greeters proceeding to the gate, GRGATE and the percentage of greeters proceeding to the parking facility and deplaning curbside for passenger pick up, PRKCRB are divided by 100 and assigned to GRGXL and PGBXL respectively. The message, 'END OF INPUT DATA' is written, and the program branches to statement number 99999, which is a RETURN.

# 4.0 MAIN SECTION

## 4.1 BAGGAGE UNLOAD SECTION

This section is called once for each arriving flight. The section first scans the matrix savevalue MH7, which was built by successive calls to 'BAGS' by the passenger transactions. Each passenger bag generates a random number from 1 to 64. The matrix MH7, which is a single column matrix of 64 rows, contains a count of the number of times each random number was generated for an arriving flight. MH7 is examined row by row in ascending order and is zeroed out after examination. For each row, the program retains a cumulative sum of bags. This cumulative sum is tested in steps of ten bags. Each time a value of ten is added to the bag count the value of the random number, which is the MH7 row number, is assigned to byte parameter number NOPB, which is initially set to forty. NOPB is then decremented by one. If a value has been assigned to byte parameter number 1 (NOPB = 1), then the value 64 is assigned to byte parameter number 1 and the program branches to 99999. This is done to insure that NOPB is not decremented to zero and then negative numbers, which would mean the program would attempt to assign a value to a byte parameter with a zero or negative number.

After all the rows of MH7 have been examined, the program determines whether the value 64 was assigned to the byte parameter which was assigned last. If this is not the case, the value 64 is then assigned to the byte parameter which was assigned last. This is done in order to cover the case

when the cumulative count of bags is not a multiple of ten. This would cause the bags in the cumulative count, after the last multiple of ten, to not have their random number assigned to a byte parameter. The assigning of 64 to the last byte parameter assures that all bags are accounted for and that all passengers with bags will be unlinked to the GPSS BAGGAGE CLAIM SECTION. The program then branches to 99999.

### 4.2 BAGGAGE CLAIM SECTION

This section is called once for each deplaning passenger who has a bag. The section first determines the MH9 row number, J, by adding the index number for baggage claim areas, INDEXF(4), to the number of the baggage claim area wanted, MH1(IV3,12). Next, the point number of the baggage claim area is determined, MH9(J,3), and placed in NPTTO. The program then assigns a statement number, 309, to NEXT which will be used to return from the WALKING TIME CALCULATION SECTION. The program then branches to the WALKING TIME CALCULATION SECTION.

After the walking time is calculated, the program branches back to statement number 309. Halfword parameter 2 is assigned the point number, NPPTO, for the baggage claim area. Byte parameter 11 is assigned the process code for the baggage claim area, 4. Halfword parameter 7 is assigned the MH9 row number, J. The program then branches to 99999.

## 4.3 CUSTOMS SECTION

This section is called once for each passenger deplaning from an international flight. The section first determines the associated customs facility number L, from MH9(IV3,4). The MH9 row number for the associated customs facility, J, is then determined by adding the associated customs facility number, L, to the index number for customs facilities, INDEXF (5). Next, the point number, NPPTO, of the associated customs facility is then assigned from MH9(J,3). The program then assigns a statement number, 313, to NEXT which will be used to return from the WALKING TIME CALCULATION SECTION. The program then branches to the WALKING TIME CALCULATION SECTION.

After the walking time is calculated, the program branches back to statement number 313. The storage and queue number, M, is determined by adding the associated facility number, L, and the base value for customs facilities, CUSQS, minus one. The subtraction is done because the variable CUSQS contains the number of the first storage for customs, thus one is subtracted in order that the number of the customs facility can be added to CUSQS in order to get the correct storage number. Halfword parameter 2 is assigned the point number, NPPTO. Halfword parameter 5 is assigned the storage queue number for customs, M. Halfword parameter 7 is assigned the MH9 row number, J, for the associated customs facility. Byte parameter 5 is assigned the process code for customs, 5. The program then branches to 99999.

## 4.4 GROUND TRANSPORTATION MODE SECTION

This section is called once for each passenger on each arrival flight by the main program and once for each passenger on a departing flight, by the auxiliary program. The section first determines if the jobtype flag, JOBT, has been set. If set, meaning that the auxiliary program is using the FORTRAN program, the program sets the variable K to 1, which indicates that the program will include the private car mode in the list of selectable transportation modes. The variable L is set to 0, and then the program determines if the array value for % preticketed, MH4(IV4,1), is less than the variable IV2. The variable IV2, which is the same as IVALUE(2), is set in the auxiliary program and passed to the FORTRAN program as the random number, RN4. If the % preticketed value is less than the random number, IV2, the flag L is set to 1, which indicates that the passenger is not preticketed. The program then goes to the next statement which is at statement number 701.

If the jobtype flag, JOBT, is not set, which means that the main GPSS program is using the FORTRAN program, the variable K is set to 3, which indicates that the program section will handle the private car mode of transportation separately from the other modes of transportation.

At statement number 701, the random number in IV3, which is the same as IVALUE(3), is normalized to a value between 0 and 1, and placed in TEMPCT. The program then determines which cumulative percentage for the different types of transportation that the normalized random number is less than or equal to, but greater than the cumulative percentage for the previous mode of transportation. The modes of transportation in their order of being tested are the following: rental car, bus/limousine, and taxi which have the codes 3, 4 and 5, respectively. If the test is satisfied for a mode of transportation then byte parameter 6 is assigned the value of J, which is the code for the mode of transportation for that passenger. Byte parameter 9 is assigned the value of L, which is the flag for whether the passenger is preticketed or not. This byte parameter is only used for this purpose in the auxiliary program and not in the main GPSS program.

If the test is not satisfied for any of the modes of transportation, that is, the normalized random number is not less than or equal to any of the cumulative percentages for the different types of transportation; the error count, NERCNT, is incremented by 1. If the error count is equal to the maximum allowable number of errors, ERRORS, then the program branches to 999. If the error count is not equal to the maximum number of errors, then the message 'PROBLEM IN GROUND TRANSPORTATION

MODE LOGIC' is written. The program then assigns byte parameter 6 the value of 4 as a default which is the code for bus/limousine. BYTE parameter 9 is then assigned the value of L. The same holds true for this assignment of byte parameter 9 as the previous assignment of byte parameter 9. The program then branches to statement number 99999.

## 4.5 RENT-A-CAR SECTION

This section is called each time a deplaning passenger goes to a car rental agency. This section first determines which rows in MH9 are car rental facilities by using the variable INDEXF(11), the index for car rental agencies in MH9, and NORENT, the total number of car rental facilities. The variable I is set to the MH9 row number which has the first car rental facility. The variable J is set to the MH9 row number which has the last car rental facility. Since each facility corresponds to a counter, several of which can belong to the same car rental agency and have the same car rental agency code number, this section must therefore scan through the car rental facilities in MH9 to determine which counter with the car rental agency code IV3 has the shortest walking time from the deplaning passenger's current position. The variable LTEMP is used to keep the car rental agency facility number. If the value in MH9(N,4), which is the car rental agency code for car rental agency facility number LTEMP, is equal to the car rental agency code, IV3, of the car rental agency that is wanted, then the program compares the walking time of that facility to the previous lowest walking time of a car rental facility with the correct car rental agency code. If the car rental facility that is being tested has a shorter walking time, then its point number is saved in MINPTO, its MH9 row number is saved in ITEMP3, and the car

rental facility number is saved in L.

After the scanning is finished, the program determines if MINPTO is greater than zero. If the variable MINPTO is greater than zero then at least one facility was found with the correct car rental process code. The program then sets the variable NPTTO to MINPTO, the point number of the car agency facility with the shortest walking distance. The statement number 326 is then assigned to the variable NEXT, and the program then branches to statement number 950 to determine the walking distance.

After the walking distance is determined, the program branches back to the statement number 326 and the program then determines the queue-storage number, M, of the car rental agency facility picked by adding the variable RCRQS to L, and subtracting one. One is subtracted because the variable RCRQS, which is passed from the GPSS program, contains the number of the first queue and storage assigned to a car rental agency facility, thus one must be subtracted from it in order to add the facility number of the car rental agency wanted to get the correct queue storage number.

The program then assigns to halfword parameter 2 the value of variable MINPTO, the point number of the car rental agency with the minimum walking distance. Halfword parameter 5 is then assigned the value of variable M, which is the queue-storage number of the car rental agency facility that was picked. Halfword parameter 7 is then assigned the value of variable

ITEMP3, which is the MH9 row number of the car rental agency facility that was picked. Halfword parameter 11 is then assigned the value 11, which is the process code for the car rental agency.

If the variable MINPTO is equal to zero then no facilities were found with the correct car rental process code. The program then scans the car rental facilities and determines if any of the car rental agency facilities have been defined. This is done by determining if the car rental agency code is greater than zero. If no car rental agencies are defined, the program checks an error flag, NRCRSW. If the error flag is equal to 1, the program branches to statement number 99999 in order not to repeat the error message. If the error flag is not equal to 1, the program sets the error flag NRCRSW to 1, and writes the message 'NO CAR RENTAL FACILITIES DEFINED. THIS MESSAGE WILL NOT REPEAT.', and branches to statement number 99999.

If, during the scan, a car rental facility is found to be defined, which means it has a car rental agency code greater than zero, then the point number, NPTTO, is obtained from MH9-(N,3), and the MH9 row number, ITEMP3, is obtained from N, then the message 'NO FACILITY DEFINED FOR CAR RENTAL AGENCY CODE,' IV3,'FACILITY FOR AGENCY CODE', K,'USED' is written, the error count NERCNT, is then incremented by 1. The program next determines if the error count is greater than the maximum allowable error count, ERRORS. If the error count is greater, the program branches to statement number 999, the section which will stop the simulation because of the cumulative error count. If the error count is not greater, the program sets the variable IV3

to K, the code for the alternate car rental agency and the variable MINPTO is set to NPTTO, the point number of the alternate car rental agency. The program then branches to statement number 326.

## 4.6 EXIT SECTION

This section is called each time a deplaning passenger is to go through an exit to the deplaning curb. The program first determines if the next address for the passenger is the deplaning curb, DPLCO, a return to the control section, CGTRO, which will immediately branch to DPLCO, or the parking garage, GRTOO. If the address is not DPLCO, CGTRO or GRTOO, then the program will set I to the value of byte parameter 1 and the message 'ATTEMPT TO EXIT TO BLOCK NUMBER', IV4, 'VIA EXIT CHECK FUNCTION', I, will be printed. The error count, NERCNT, is incremented by 1, and then tested to determine if it is equal to the maximum allowable number of errors, ERRORS. If NERCNT is equal to ERRORS, the program branches to 999, the section which will stop the simulation because of the cumulative error count. If NERCNT is not equal to ERRORS, the program branches to 99999.

When the address is either DPLCO, CGTRO, or GRTOO, the program determines if the current process, IV3, is for a gate, baggage claim, customs, rent-a-car, or security which have process numbers 1, 4, 5, 11, or 3, respectively. The program branches to the part of this section corresponding to the current process number. Regardless of the process number, each program section that is branched to has the same logic. The reason for this is so that any future changes for one type of facility program section could be easily modified without

interfering with the logic for the other types of facilities. The logic for each type of facility is as follows:

The variable J is equated to the value of MH9 (IV5,3), which is the point number of the passenger's current location. The index, IV5, is the MH9 row number of the last facility. The variable NPTTO is then set to the value of MH3 (J,3), the point number of the exit closest to the facility. The statement number following the next instruction is assigned to the variable NEXT. The program branches to statement number 950 to determine the walking time to the exit. After the walking time is determined, the program branches back to the statement following the 'GO TO 950' statement. Halfword variable 2 is then assigned the value of NPTTO, the point number of the nearest exit. The program then branches to 99999.

## 4.7 IMMIGRATION SECTION

This section is called for each deplaning passenger from an international flight. The variable NPTFM is set equal to IVALUE(2), the point number of the current location. The variable IV3 is set to IVALUE(3), the gate number. The variable L is then set to MH9(IV3,5), the number of the designated immigration facility for that gate. Gate index numbers do not need to be determined because the gate facilities are the first type of facility in MH9, and their index number would be zero. If L is greater than zero then the designated immigration facility for that gate has been defined, and the program branches to statement 335 to continue normal processing.

At statement number 335 the variable J is set to L plus INDEXF(13), the index number for immigration facilities. This determines the MH9 row number for the immigration facility specified. The variable NPTTO is set to MH9(J,3), the point number of the specified immigration facility. Statement number 338 is assigned to the variable NEXT, and the program branches to statement number 950 to determine the walking time.

After the walking time is calculated, the program branches back to statement number 338. The variable M is then set to IMMOS plus L minus 1 where IMMOS, which is passed from the GPSS program, is the number of the first queue-storage used for immigration facilities. M is then the number of the

queue-storage associated with immigration facility number L. The reason for subtracting 1 from L is the same for the setting of the variable M in the RENT-A-CAR SECTION, Section 4.5. Halfword parameter 2 is then assigned the value of NPTTO, the point number of the designated immigration facility. The queue storage number, M, is placed in halfword parameter 5. Halfword parameter 7 is set to the value of J, the MH9 row number. Byte parameter 11 is assigned the value 13, which is the process code for immigration facilities. Halfword parameter 8 is also set to J, the MH9 row number, so that the MH9 row number of the immigration facility can be passed back to the FORTM program from the Customs Section of the GPSS program. The value of J in PH7 is lost before the transaction gets to the Customs Section of the GPSS program. The program then branches to statement number 99999.

If the value of L is not greater than zero then one designated immigration facility has been defined for that gate and the program starts checking for errors. The program then determines if the variable NOIMMI, which is the number of immigration facilities, is greater than zero. If NOIMMI is not greater than zero, then no immigration facilities have been defined and the program writes the message, 'PASSENGER ATTEMPTED TO GO TO IMMIGRATION. NO FACILITIES DEFINED'. The error count, NERCNT, is incremented by one and the program determines if the error count is equal to the maximum allowable number of errors, ERRORS. If the error count is equal to ERRORS then

the program branches to statement number 999, the section which will stop the simulation due to the cumulative error count. If the error count is less than ERRORS, the program branches to statement number 99999.

If the variable NOIMMI is greater than zero then there is at least one defined immigration facility, even though the designated immigration facility for the particular gate is not specified. The variable J is set to INDEXF(13), the index number for immigration facilities. The variable K is set to J plus NOIMMI, to obtain the MH9 row number of the last immigration facility. J is then incremented by 1 to obtain the MH9 row number of the first immigration facility. The program then tests each immigration facility in turn, keeping the variable L as the number of the facility, to determine the first immigration facility that has a point number, MH9(N,3), which is greater than zero, indicating that the facility has been defined. L is then set to the point number of the chosen immigration facility. The message, 'NO IMMIGRATION FACILITY DEFINED FOR GATE', IV3, L, 'CHOSEN', is then written. The error count, NERCNT, is then incremented by one. The program then determines if the error count is equal to the maximum allowable error count, ERRORS. If the error count is equal to ERRORS, then the program branches to statement number 999. If the error count is less than ERRORS, the program continues to the next statement which is at statement number 335.

## 4.8 PASSENGER DEPLANING CURB SECTION

This section is called once for each deplaning passenger proceeding to the deplaning curb. The variable IV2 and IV3 are set to NPTFM and IVALUE (3) which are the respective current process code and the previous facility number for facilities other than an exit. The variable IV5 is set to IVALUE (5), the flight table row number. The program determines if the current process code, IV3, is for a gate baggage claim, customs, rent-a-car, or check-in, which have process codes of 1, 4, 5, 11, and 14 respectively, and are the only facility types that would send a passenger to curbside. If the current process code is not one of these facility types then the program starts an error processing procedure. The variable I is set to byte parameter 1, which is the process function number. The message, 'ATTEMPT TO EXIT TO DEPLANING CURB FROM FACILITY TYPE', FACTYP (IV3), 'CHECK FUNCTION', I, is written. The error count, NERCNT, is then incremented by one and tested against the maximum allowable number of errors, ERRORS. If the error count is equal to ERRORS, the program branches to statement number 999. If the error count is less than ERRORS, the program branches to statement number 99999.

If the current process code, IV3, is 1, for a gate facility, then the program branches to statement number 600 where the variable I is set to MH1 (IV5, 12), which is the baggage claim area number specified for that flight, plus INDEXF(4), the index number for baggage claim areas.

This obtains the MH9 row number for the specified baggage claim area. The variable ITEMP1 is then set to MH9(I,4), the deplaning curb facility number for that baggage claim area. The program then branches to statement number 690.

If the current process code, IV3, is 4, which is for a baggage claim area facility, then the program branches to statement number 605 where the variable I is set to IVALUE(4), which is the MH9 row number for the previous facility. The variable ITEMP1 is then set to MH9(I,4), which is the deplaning curb facility number for that baggage claim area. The program then branches to statement number 690.

If the current process code, IV3, is 5, for the customs facility, the program then branches to statement 610 where the variable I is set to IVALUE(4) which is the MH9 row number for the previous facility. The variable ITEMP1 is then set to MH9(I,4), which is the deplaning curb associated with the Customs facility. The program then branches to statement number 690.

For the current process code, IV3, equal to 11, the car rental facility, the program branches to 615 where ITEMP1 is set to MH9(I,5), the parking facility number associated with the rent-a-car agency. The program then branches to statement 690.

When IV3 is 14, the transaction currently processed represents a deplaning passenger without baggage and will either be met by greeters at curbside or will use a bus or taxi. This passenger is routed to the airline check-in facility and then

to the enplaning curb. At statement 620 the program obtains the airline number from MH1(IV5,3). The corresponding enplaning curbside number is obtained from MH2(I,1) and the facility number J, for the enplaning curbside is obtained by adding INDEXF(8) to this. The program then branches to 692.

At statement number 690, the variable J is set to the variable ITEMP1 plus INDEXF(12), the index number for the deplaning curb facility specified. The point number of the deplaning curb is placed in NPTTO at statement 692. The program then assigns the statement number 691 to the variable NEXT. The program branches to statement number 950, where the walking time is determined.

After the walking time is determined the program returns to statement number 691. Halfword parameter 2 is then assigned the value of NPTTO. Halfword parameter 7 is assigned the value of J, the MH9 row number of the deplaning curb area. Byte parameter 11 is assigned the value 12 which is the process code for a deplaning curb. The program then branches to statement number 99999.

4.9 CAR DEPLANING CURB SECTION

This section is called by greeter transactions for passengers to be met at curbside and those who have met passengers inside the terminal. It assigns transactions to curbside, double parking, or queue areas dependent upon current congestion.

The variable IV2 is set to IVALUE(2), the airline number. IV3 is set to IVALUE(3), the MH1 row number, and IV4 is assigned IVALUE(4), the number of bags of the transaction representing the terminating deplaning passenger. For IV4 not equal to zero the program branches to 700.

When IV4 is 0, indicating no bags, the greeter transaction is routed to the enplaning curb for passenger meeting. The number of the enplaning curb, MH2(IV2,1), assigned to the airline is placed in the variable M. If IVALUE(5) equals one, indicating a greeter that has recirculated and parked, the program branches to 716.

The program then performs a curb search for an open space. For a fixed value of M, the matrix IEPSCH(K,M), provides the sequence of enplaning curbside numbers to be searched for an open space. A DO loop, ending at statement 713, with a range from 1 to 10 for the index K, executes this search. The variable L is set to IEPSCH(K,M) and first tested to determine if it exceeds the number of input enplaning curbside facilities, NOENPL. Values of L greater than NOENPL are skipped by branching to 713.

Allowable values of L are added to INDEXF(8) to determine the facility number ITEMP1. To determine if this facility has been input, the program tests MH9(ITEMP1,3) for zero. If undefined, this facility number is skipped. For valid facility numbers the program calculates the storage number J from EPCBS + L-1. To examine the availability of the curbside storage, the subscript ITEMP3 is calculated using the expression 11*(J-1)+2. The GPSS reference word ISTO(ITEMP3) is tested. When the value ISTO(ITEMP3) is zero, indicating no enplaning curbside spaces available, the program branches to statement 714 to begin searching for a double-parking slot at the same curbside. If a non-zero availability value is present, the value J is assigned to PH6 and PB10 is set to 1 indicating a curbside parking location. The program branches to 99999 for a return to GPSS.

At statement 714 the storage number, J, is calculated from EPDPS + L-1. The subscript ITEMP3 is again calculated by the expression 11*(J-1)+2. The availability of a double parking slot is tested. If found, the value J is assigned to PH6 and PB10 is 2. The program branches to 99999 and returns to GPSS.

When no parking is available at curbside or in a double parking slot, the program examines the next enplaning curbside area indicated by the matrix IEPSCH(K,M). When all possible areas have been tried and no space is available the program attempts to find a queue space at the enplaning curb M of the IVALUE(2) airline. The storage number, J, of this

queue is calculated from EPQCS + L-1, where L is equal to M. The subscript ITEMP3 is evaluated by using 11*(J-1)+ 2 as before. The storage representing the queuing at curbside is tested for availability. If a slot is found, J is assigned to PH6 and PB10 is set to 3. The program branches to 99999 and returns to GPSS. When there are no available queue slots, the car must recirculate. The parameter PH6 is set to zero and PB10 is set to 4. The latter value indicates that the transaction must proceed to the recirculation road section of the GPSS program. The program branches to 99999 and returns to GPSS.

The greeter accompanying a passenger without bags, who has recirculated, parked and then proceeds from parking to enplaning curbside, obtains a facility number J at statement 716. Parameter byte 11 is assigned the value 8. The program branches to 718 where this transaction will be further processed with those having passengers with baggage.

Greeters accompanying passengers with baggage are routed to the deplaning curb logic of this section beginning at statement 700. The facility number I is obtained by adding MH1(IV3,12), the bag claim area assigned to the flight and INDEXF(4), the index value for bag claim facilities.

If IVALUE(5) equals one, indicating a greeter that has recirculated and parked, the program branches to 717. For transactions performing initial processing at the curb, the storage number J of the deplaning curbside associated with the bag claim area facility number I is obtained from MH9(I,4)

+ DPCBS-1. At this curbside, the program tests for availability at curbside, or, if necessary, for double parking availability using the same logical structure as the enplaning curbside. The procedure here differs slightly since only the single assigned curbside and associated double parking area are examined and the variables DPCBS and DPDPS are used in place of EPCBS and EPDPS, respectively. When no space is found at the deplaning curbside or double parking area, the program branches to statement number 711 to test the availability of a queue space without searching other curbside areas. The storage number J of the deplaning curbside is obtained from MH9(I,4) + DPQCS-1, and the GPSS subscript number, ITEMP3, is obtained from 11*(J-1)+2 to test the availability of the storage. If no space is available, the car must recirculate. Each of the above conditions cause branching to 99999 and return to GPSS.

At 717, greeters returning to the deplaning curb from parking are assigned the facility number J of the curbside associated with bag claim area, I, from the expression MH9 (I,4) + INDEXF(12). A value of twelve is assigned to PB11 to indicate curbside as the current process code. The travel time from parking to curbside is calculated. The point number of the deplaning curbside is assigned to PH2 and the facility number J is assigned to PH7. The program returns via branching to 99999.

4.10 ENPLANING CURB SECTION

This section is called for each originating enplaning passenger transaction using private car, taxi, or bus for ground transportation. The section first sets the following variables: IV2 to IVALUE(2), the airline number; IV3 to IVALUE(3), the transportation mode; and J to MH2(IV2,1), the enplaning curb facility number for airline number IV2. The program then branches according to the mode of transportation indicated by IV3.

When the value of IV3 is 1, or 5, which is for private car or taxi drop-off, the program searches through the array IEPSCH, for each enplaning curb facility, which contains the order that the enplaning facilities are to be examined in order to find an open curb space for the vehicle. The search scheme always first determines if the enplaning curb facility specified by the airline has an open curb space before trying the other enplaning curb facilities. The variable L is set to IEPSCH(K,J),(K will vary from 1 to 10) the enplaning curb facility to be tested for an open space. The program then determines if L is greater than NOENPL, the number of enplaning curb facilities, indicating that enplaning curb facility number L is undefined. If it is undefined the program tests the next enplaning curb facility as specified in array IEPSCH for curb facility number J. If enplaning curb facility L is defined then the variable ITEMP1

is set to INDEXF(8) plus L, which gives the MH9 row number for the enplaning curb facility L. The program then determines if MH9(ITEMP1,3) is equal to zero, indicating that the enplaning curb facility is a dummy facility. When a dummy facility is encountered, the program tests the next enplaning curb facility. If the enplaning curb facility is not a dummy facility, the program sets M to EPCBS plus L minus one where EDCBS, which is passed from the GPSS program, is the number of the first storage used for enplaning curb facilities. M is thus the number of the storage associated with enplaning curb facility number L. The reason that one is subtracted from EPCBS is the same as for the setting of the variable M in the RENT-A-CAR SECTION, Section 4.5. The variable ITEMP3 is then set to 11*(M-1)+2, the subscript for the number of available units in storage number M. The program next determines if the number of available units in storage number M is equal to zero, indicating no open space at the enplaning curb. If there is not a free space at the curb, the program branches to statement number 804. When a space is available, the storage number M is assigned to PH6 and PB10 is set to 1, indicating an assignment to curbside. The program branches to 803.

At 804, the storage number M, for double parking at curb L, is determined from EPDPS + L-1. The subscript ITEMP3 is calculated from 11*(M-1)+2. The storage M availability is tested for a zero value, indicating no open space at the

double parking area of curb L. If no space is available, the program branches to statement 800 and continues the curb search loop. When double parking is available, the program assigns to PH6 and 2 to PB10, flagging an assignment to double parking. The program branches to 803.

If all the enplaning curb facilities have been tested and no curbside or double parking space is found, the program attempts to locate a space in the queue adjoining the airline enplaning curb facility. The enplaning curbside facility number J is assigned to L. Facility number ITEMP1 is calculated from INDEXF(8)+L. The storage number M is calculated from EPOCS+L-1. The subscript, ITEMP3, as before, is calculated from 11*(M-1)+2. The storage M availability is tested for a zero value, indicating no space for queuing at the enplaning curbside. If no space is available, the program branches to 805 to provide recirculation.

When a queue space is available, M is assigned to PH6 and 3 to PB10, as a flag for queuing for a parking space. The program branches to 803 to calculate the point number of the enplaning curb.

At 805, vehicles which must recirculate are assigned zero to PH5 and PH6. A flag value of 4, indicating recirculation, is assigned to PB10. The program branches to 99999.

Vehicles assigned to curbside, double parking, or queuing, use statement 803 where the point number of the curbside is

determined from MH9(ITEMP1,3). The point number, NPTTO, is assigned to PH2 and facility number ITEMP4 is assigned to PH7. The program branches to 99999 and returns.

If the value of IV3 is 5, which is for bus/limousine service, the program sets ITEMP2 to MH2(IV2,3), the enplaning curb facility number for a bus stop for airline number IV2. If ITEMP2 is greater than zero, indicating that the enplaning curb facility number for bus/limousine service is different from the private car enplaning curb facility number for that airline, then the program branches to statement number 809. If ITEMP2 is not greater than zero, then the enplaning curb facility number for bus/limousine service is the same as the private car enplaning curb facility number and ITEMP2 is set to MH2(IV2,1), the private car enplaning curb facility number. At the next statement, which is statement number 809, the program sets ITEMP1 to INDEXF(8) + ITEMP2, the MH9 row number for enplaning curb facility number ITEMP2. The variable NPTTO is then set to MH9(ITEMP1,3), the point number for the enplaning curb facility. Halfword parameter 2 is assigned the value of NPTTO, the point number, and halfword parameter 7 is assigned the value of ITEMP1, the MH9 row number for the enplaning curb facility. The program then branches to statement number 99999.

4.11 ENTRANCE SECTION

This section is called each time an enplaning passenger or vistor comes to an entrance. The variable NPTFM is assigned the value IVALUE(2), the point number of the current location. The variable NPTTO is set equal to MH3(NPTFM,4) which is the point number of the nearest entrance. Statement number 813 is assigned to the variable NEXT. The program then branches to statement number 950 to determine the walking time.

After the walking time is calculated, the program branches. back to statement number 813. Halfword parameter 2 is assigned the value of NPTTO, the point number of the entrance. The program then branches to statement number 99999.

## 4.12 TICKETING AND CHECK-IN SECTION

This section is called for enplaning passengers not proceeding directly to security, for deplaning passengers exiting the terminal building without bags, and for greeters. The program first sets NPTFM to IVALUE(2), the point number of the current location, and IV3 to IVALUE(3), the airline code number. The program tests PB8 for 1, indicating a deplaning passenger, and branches to 844 for this passenger type. It also tests for greeters routed to ticketing for meeting deplaning passengers without bags and branches to 844 for this group. Enplaning passengers are tested for a non-preticketed status, IVALUE(4) equal to 1, or if the random number, in IVALUE(5) from RN4 in the GPSS program, is greater than the percentage of preticketed passengers using the express check-in facility, MH2(IV3,2). If the test is true, the program branches to the area for express check-in facilities which starts at statement number 850. Otherwise, the program continues to statement number 844.

The full service facility area starts with J set to INDEX(14), the index number for full service ticket facilities. Next K is set to J+NOTICK to obtain the last MH9 row number for full service ticket facilities. J is then incremented by 1 to obtain the first MH9 row number for full service ticket facilities. The program then searches through the full service ticket facilities to find the one that has the same airline code as the passenger with the facility number saved in L. If there is a match of airline codes between the passenger and the

full service ticket facility, the program branches to statement number 848. If there is no match, the program enters an error processing area for undefined full service ticket facilities.

In the error processing area, the program first determines if NOTICK, the number of full service ticket facilities, is greater than zero. If it is not greater than zero, the program writes the error message, 'NO TICKETS & CHECKIN FACILITIES DEFINED FOR ENPLANING PASSENGERS. RUN TERMINATED,' and the program then branches to statement number 999. If it is greater than zero, the program will use the first full service ticket facility. The variable L is set to 1 to indicate that facility number. The variable I is set to INDEXF (14) + 1, the MH9 row number for the first full service ticket facility. The variable N is set to MH9(I,4) the airline code for the first full service ticket facility. The program then writes the message, 'NO TICKET & CHECKIN FACILITY DEFINED FOR AIRLINE CODE', IV3, 'FACILITY OF AIRLINE CODE', N, 'USED'. The error count, NERCNT, is incremented by one, and the program determines if it is equal to ERRORS, the maximum allowable error count. If it is not equal to ERRORS the program goes to the next statement which is statement 848.

At statement number 848 the program sets M to TICQS + L - 1, where TICQS, passed from the GPSS program, is the number of the first queue-storage associated with the full service ticket facility. This obtains the queue-storage number for full service ticked facility number L. One is subtracted from M

for the same reason that one is subtracted from M in the RENT-A-CAR SECTION, Section 4.5. The variable ITEMPL is then set to CHECK3. This variable is passed from the GPSS program and contains the number of the block location which the GPSS program will branch to for full service ticket facilities. N is then set to 14, which is the processing code for full service ticket facilities. The program then branches to statement number 857.

The express check-in facility area, which starts at statement number 850, first sets J to INDEXF(2), which is the index number for express check-in facilities. Next, K is set to J + NOCHEC, where NOCHEC is the number of express check-in facilities, to obtain the MH9 row number of the last express check-in facility. J is then incremented by one to obtain the MH9 row number of the first express checkin facility. The program searches through the airline codes for the express checkin facility in MH9(I,4) to determine which facility has the same airline code as the passenger. The number of the express checkin facility with the same airline code as the passenger is saved in variable L. If there is a match, the program branches to statement number 853. If there is no match, the program enters an error processing area and will attempt to use any full service facility.

In the error processing area the program first sets J to INDEXF(14), the index number for full service ticket facilities. K is then set to J + NOTICK to obtain the MH9 row number of the last full service ticket facility. J is then incremented

by 1 to obtain the MH9 row number for the first full service ticket facility. The program then searches through the airline codes for the full service ticket facilities, contained in MH9(I,4), to determine which facility has an airline code that matches with the passenger's airline code in IV3. The number of the matching facility is saved in the variable L. If there is a match, the passenger will be sent to that full service facility and the program will branch to statement number 859. If there is no match, the program first determines if NOTICK, which is the number of full service ticket facilities, is greater than zero, indicating that at least one full service facility has been defined. If NOTICK is not greater than zero then the message, 'NO TICKET & CHECKIN DEFINED FOR ENPLANING PASSENGERS. RUN TERMINATED', is printed and the program branches to statement number 999. If NOTICK is greater than zero the program sets I to INDEXF(14) + 1 to obtain the MH9 row number of the first full service facility. Next, N is set to MH9 (I,4) to obtain the airline code for the first full service ticket facility. The message, 'NO EXPRESS CHECKIN FACILITY DEFINED FOR AIRLINE CODE', IV3, 'FULL SERVICE AIRLINE CODE', N,'USED', is then written. The error count, NERCNT, is incremented by 1 and L is set to 1 for the number of the first full service ticket facility. If the error count is equal to ERRORS, the maximum allowable number of errors, the program branches to 999. If the error count is not equal to ERRORS, the program goes on to the next statement which is statement number 859.

At statement number 859, M is set to TICQS + L -1, where TICQS is number of the first queue-storage associated with full service ticket facilities. This obtains the queue-storage number for facility number L. One is subtracted from TICQS for the same reason that one was subtracted from M in the RENT-A-CAR SECTION, Section 4.5. Next, ITEMP1 is set to CHEK3. This variable is passed from the GPSS program and contains the number of block location which the GPSS program will branch to for full service ticket facilities. N is then set to 14, which is the process code for full service ticket facilities. The program then branches to statement number 857.

The following statement, which is at statement number 853, continues the processing for express check-in facilties by setting M to CHKQS -1 + L, where CHKQS is the number of the first queue-storage associated with express check-in facilities. This obtains the queue-storage number for express check-in facility Number L. One is subtracted from CHKQS for the same reason as above. N is next set to 2 which is the process code for express check-in. ITEMP1 is then set to CHEK2. This variable is passed from the GPSS program and contains the number of the block locations which the GPSS program will branch to for express checkin facilities. The program then branches to statement number 857 which is the following statement.

At statement 857 the program sets NPTTO to MH9(I,3) to obtain the point number of the full service or express checkin facility. The statement number 856 is assigned to

NEXT and the program branches to 950 to determine the walking time.

After the walking time is determined, the program branches back to statement number 856. The program then assigns to halfword parameter number 2 the value of NPTTO, the point number of the full service or express checkin facility. Halfword parameter 4 is then assigned the value of ITEMP1, the block location that the GPSS program will branch to for either full service or express check-in facilities. Halfword parameter 5 is assigned the value of M, the queue-storage number for the full service or express checkin facility. Halfword parameter 7 is assigned the value of I, the MH9 row number for the full service or express checkin facility. Byte parameter 11 is assigned the value of N, the process code for full service or express checkin facilities. The program then branches to statement number 99999.

## 4.13 SECURITY SECTION

This section is called for each enplaning passenger and greeters proceeding to the gate. The variable NPTFM is set to IVALUE(2), the point number of the current location; and IV3 is set to IVALUE(3), the number of the gate the passenger is proceeding to. I is then set to MH9 (IV3,4), the security facility number for gate number IV3. I is then tested to determine if it is greater than zero. If I greater than zero then the security facility has been defined for that gate, and the program branches to statement number 860. If the value of I is not greater than zero then the program writes the message, 'NO SECURITY FACILITY DEFINED FOR GATE', IV3, 'SECURITY FACILITY NUMBER 1 IS ASSIGNED'. MH9(IV3,4) and I are set to 1 in order to assign security facility 1 to gate number IV3 for current and future reference.

At the following statement, which is statement number 860, J is set to INDEXF (3) +I the MH9 row number of security facility number I. M is set to SECQS+I-1, where SECQS is the number of the first GPSS queue-storage associated with security facilties, to obtain the queue-storage number for security facility number I. One is subtracted from SECQS for the same reason that one is subtracted from M in the RENT-A-CAR SECTION, Section 4.5. NPTTO is next set to MH9 (J,3), the point number of the security facility. Statement number 861 is assigned to NEXT, and the program then branches to statement number 950 to determine the walking time.

After the walking time has been calculated the program branches back to statement number 950. The program then assigns to halfword parameter 2 the value of NPTTO, the point number of the security facility. Halfword parameter 5 is next assigned the value of M, the queue-storage number for the security facility. Halfword parameter 7 is assigned the value of J, the MH9 row number of the security facility. Halfword parameter 11 is then assigned the value of 3, the process code for security. The program then branches to statement number 99999.

## 4.14 GATE SECTION

This section is called for each enplaning passenger and greeters proceeding to the gate. The variable NPTFM is set to IVALUE(2), the point number of the current location; and IV3 is set to IVALUE(3), the number of the gate the passenger is proceeding to. NPTTO is then set to MH9(IV3,3), the point number of the gate. No index number is needed to access the gate information in MH9, since the gates are the first facility type in MH9 and the index number would be zero. The program then determines if NPTTO is greater than zero which would indicate that the gate has been defined and is not a dummy facility. If NPTTO is greater than zero, the program branches to statement number 873.

If NPTTO is not greater than zero, indicating that the gate is a dummy facility, the program scans through the gate facilities in MH9 to find a gate that is not a dummy facility, indicated by MH9 (I,3) being greater than zero, where I is the number of the gate being tested. When a non-dummy gate facility is found the program sets J to halfword parameter 1, which is the flight table row number for the flight that the passenger is going to take. The gate number for the flight MH1(J,9) is then set to I so that all subsequent passengers for that flight will go to gate number I. The message, 'GATE', IV3, 'NOT DEFINED. CHECK DATA FOR FLIGHT', MH1(J,2), 'GATE', I, 'USED', is written and IV3 is set to I, the new gate number. NPTTO is then set to MH9(IV3,3), the point number of the new gate.

The following statement, which is at statement number 873, assigns statement number 874 to NEXT. The program then branches to statement number 950 to determine the walking time. After the walking time is determined the program branches back to statement number 874. Next, M is set to GAQSL+IV3-1 where GAQSL, which is passed from the GPSS program, is the number of the first queue-storage associated with the gate facilities. This obtains the number, M, of the queue-storage for gate facility number IV3. One is subtracted from GAQSL for the same reason that one is subtracted from M in the RENT-A-CAR SECTION, Section 4.5.

Next, halfword parameter 2 is assigned the value of NPTTO, the point number of the gate. Halfword parameter 5 is assigned the value of M, the queue-storage number for the gate. Halfword parameter 7 is assigned the value of IV3, the MH9 row number for the gate, which in the case of gate facilities is the same as the number of the gate. Byte parameter 11 is assigned the value of 1, which is the process code for gates. The program then branches to statement number 99999.

## 4.15 PARKING SECTION

This section differs from other FORTM sections because it is called from several locations in the GPSS program. Furthermore, transactions with four different requirements call the parking section.

These requirements, and the types of transactions utilizing them are the following:

(1) Parameter assignments to specify the queue storage numbers for subsequent simulation of parking lot exits

- used by deplaning passengers, either self-driven or with accompanying greeters.
- used by well-wishers departing the airport.

(2) Parameter assignments to specify the point number of the parking facility and the parking lot number.

- used by enplaning passengers self driven or with well wishers.
- used by enplaning passengers returning rental cars
- used by greeters meeting passengers inside the terminal.

(3) Parameter assignments to specify point number and queue storage number of parking lot exit

- used by greeters proceeding from parking lot to curb.

(4) Parameter assignment to specify parking lot number

- used by well wishers proceeding from enplaning curb to parking lot.

The program first sets NPTFM to IVALUE(2), the point number of the current location; IV3 to IVALUE(3), the transportation mode; IV4 to IVALUE(4), the deplaning/enplaning flag (0.1); IV5 to IVALUE(5), the car rental agency number; and, IV6 to IVALUE(6), a flag to signify that only the lot number will be obtained. The program then determines if the transaction represents a passenger or well-wisher by testing IV4 for a value of 1. If the transaction represents either category the program branches to statement number 720. If the passenger is deplaning or a greeter is represented the program determines if the passenger or greeter is driving a private vehicle or renting a car, and branches to statement number 728 or 722, respectively. If the transportation mode is neither of these the program branches to statement number 721 where an error processing area starts.

At statement number 720 the program determines if the enplaning passenger or well-wisher is driving a private vehicle or renting a car and branches to statement number 728 or 722, respectively. If the transportation mode is neither of these the program goes to the following statement, statement 721, where an error processing area starts. The variable I is set to halfword parameter 4, the address parameter. The program then writes the message, 'INVALID CALL TO FORTM PARKING. 'PH2=', NPTFM, 'PH4=' I, 'PB7=', IV4, 'PB6=' IV3. The error count,

NERCNT, is incremented by one and is compared with the maximum allowable error count, ERRORS. If NERCNT is equal to ERRORS, the program branches to statement number 999. If NERCNT is not equal to ERRORS, the program branches to statement number 99999.

The following statement, which is at statement number 722, sets I to INDEXF(11), the index number for car rental facilities. J is then set to I+NORENT, which is the last MH9 row number for car rental facilities. I is incremented by 1 to make it the first MH9 row number for car rental facilities. The program then scans the car rental agencies and compares the agency code of each car rental facility in MH9(N,4) with the agency code of the passenger. When a match is found, L is set to MH9(N,5), where N is the MH9 row number for car rental facilities, to obtain the parking lot facility number for that car rental facility. If L is greater than one, indicating that the parking lot is a special lot for the rental agency, the program branches to statement number 723. If L is equal to one, indicating that the general parking lot is used by the car rental facility, the program continues scanning the car rental facilities.

If no match of agency codes between car rental facilities and passenger is made with the parking lot facility number being greater than one, the program continues to the following statement, which is statement number 728. At statement number 728 a lot number, LOTNO, is obtained from PB14 if one was previously

assigned. For those without this assignment, LOTNO is given a value of 1 which assigns the transaction to the general lot. The facility number N is then set to INDEXF(10) + LOTNO, the MH9 row number for the specified parking lot. M is set to PARQS + LOTNO-1, where PARQS, passed from the GPSS program, is the number of the first queue-storage associated with parking lot facilities. The program tests IV6 for a value of 1, to determine if only the lot number is required. For other values, the program branches to 724. The lot number is assigned to PB14, and the program branches to 99999 for a return to GPSS.

At the following statement, which is at statement number 723, the program sets N to INDEXF(10) + L to obtain the MH9 row number for parking lot facility number L. M is then set to PARQS+L-1 to obtain the queue-storage number for parking lot facility number L One is subtracted from PARQS for the same reason that one is subtracted from M in the RENT-A-CAR SECTION, Section 4.5.

At statement number 724, NPTTO is set to MH9(N,3), the point number of the parking lot facility. If NPTFM is zero, for an enplaning passenger or greeter, the program branches to statement number 727 to skip the walking time determination since the parking lot was the first landside facility used, and no walking time determination is needed. Otherwise the program assigns statement 727 to NEXT and branches to statement number 950 to determine the walking time.

After the walking time is calculated, the program branches back to statement number 727. Halfword parameter 2 is set to NPTTO, the point number of the parking lot facility. Halfword parameter 5 is set to M, the queue-storage number of the parking lot facility. Halfword parameter 7 is set to N, the MH9 row number of the parking lot facility. Byte parameter 11 is set to 10, the process code for parking lots, and byte parameter 14 is set to LOTN. The program then branches to statement number 99999.

4.16 TRANSFER PASSENGER SECTION

This section is called once for every transfer and transit passenger. Transfer passengers are those arriving and departing at different gates. Transit passengers arrive and depart at the same gate. The program first sets M to IVALUE(5), the arriving gate number. ITEMP3 is next set to MH9(M,4), the security facility number (and concourse number) for that gate. Next, ITEMP3 is tested to determine if it is greater than zero, which indicates that a security facility has been assigned to the gate. If ITEMP1 is greater than zero, the program branches to statement number 827. If ITEMP1 is equal to zero, the program writes the message, 'NO SECURITY FACILITY FOR GATE', M, 'SECURITY FACILITY NUMBER 1 ASSIGNED'. MH9 (M,4) and ITEMP3 are both set to 1 to assign security facility 1 to gate IVALUE(5).

Statement 827 places IVALUE(2) in IV2. The program executes a computed GO TO and branches to program statements 821 or 822 if the passenger is a transfer passenger or transit passenger, respectively. The following statement, which is statement number 821, determines if NOFXFR, the number of available transfer flights, is greater than zero. If NOFXFR is greater than zero, indicating there are transfer flights available, the program branches to statement number 824. If NOFXFR is equal to zero the program sets K to PB5, the number in the party. MH11(ITEMP3) is then incremented by K to keep count of passengers that leave concourse number ITEMP1. The save-value XFRXH is incremented by 1 to keep

count of the number of transfer transactions that are not accepted on a transfer flight. The block location TRX99 is assigned to PH4, and the block location CTRL1 is assigned to PH8. When the FORTRAN program returns to the GPSS program, these two assignments will cause the transfer transaction to terminate. The program then branches to statement number 99999.

The following statement, statement number 824, assigns block location CTRLO to PH8, which will cause the GPSS program to process the transaction normally once the FORTM program returns. The variable N is set to the remainder of IVALUE(3), which is a random number passed from the GPSS program, divided by NOFXFR, the number of available transfer flights, plus 1. This will cause N to be assigned a random integer between 1 and NOFXFR, which will be used as the row number of the transfer flight matrix MH5 for this transfer passenger. The variable I is then set to MH5(N) to obtain the MH1 row number of the transfer flight. The variable K is next set to the address for MH1(I,11). The quantity stored at this address is the number of seats still available for transfer passengers. MH1(I,11) is then decremented by 1 to indicate that another seat has been occupied by a transfer passenger.

The following statement determines if MH1(I,11) is greater than zero or not. If it is greater than zero, indicating that there are transfer passenger seats available on the flight, the program branches to statement number

820. If MH1(I,11) is equal to zero, indicating that all transfer seats are taken, the program deletes the flight number in row N from the transfer flight table MH5 by moving all flights in MH5, after flight number N, one row closer to the beginning of the matrix. The number of transfer flights, NOFXFR, is decremented by 1.

At the following statement, which is at statement number 820, the program assigns the MH1 flight table row number, I, to PH7, branches to 99999 and returns to GPSS. The transit passenger's arriving flight table row number is contained in IVALUE(3). At statement 822 this is assigned to K. The gate number, IGAT, of this flight is determined from MH1(K,9). The flight table matrix MH1 is examined starting from row K+1 to determine the next departure at the same gate. MH1(I,1) is tested in a DO LOOP from I=K+1 to I=999 for negative, zero, and positive values. If negative, indicating the end of the flight table matrix, the program branches to statement 818. If zero, indicating an arriving flight, the program branches to 826 to search the next row, If positive, indicating a departing flight, the program branches to 819. At 819 the gate number in MH1(I,9) is compared to IGAT. If these are identical the program branches to 817. If not identical the program continues to statement 826 to continue the search.

At statement 818, which follows statement 826, the number of transit passengers in the party, contained in byte parameter 5, is assigned to K. Matrix MH11 is

incremented by K. Although this passenger was intended to act as a transit passenger, no matching gate number was found and this passenger is included in the count XFRXH of transfer passengers unable to obtain a connecting flight. The value XFRXH is incremented by K. The passenger transaction is assigned TRX99 to PH4 and CTRL1 to PH8 for termination upon return to the GPSS program.

Transit passengers successfully obtaining a matching gate number are routed to statement 817. Flight table row number, I, is assigned to PH1 and PH8 is assigned CTRL1 for transfer to the next point in the transit passenger routing function.

The program branches to 99999 for a return to GPSS.

## 4.17 TRANSFER FLIGHTS SECTION

This section is called at the start of the simulation to initialize the flight table and later called to add to or delete a flight from the flight table. If transfer seats remain unfilled when the flight is to be deleted, this section is called to assign point numbers to transactions created to complete the count of transfer passengers. Only the transaction representing the flight performs this call.

The variable IV2 is first set to IVALUE(2), the MH1 row number. IV3 is next set to IVALUE(3), the flag indicating flight table initialization, addition, deletion, or point number assignment. The program next tests if IV3 is equal to 1, the flag setting for deleting a flight from the flight transfer table. If IV3 is equal to 1, the program branches to statement number 832. If IV3 is not equal to 1, the program tests if IV3 is equal to 2, the flag setting for adding a flight to the flight transfer table. If IV3 is equal to 2, the program branches to statement number 830. If IV3 is equal to 3, the program branches to 836 for point number assignment. If IV3 is none of the above, the flight transfer table is to be initialized.

The program then tests each flight, I, in MH1, which is the flight table matrix, in several different ways. The first test determines whether MH1(I,1) is negative, zero, or positive. This flag tells whether the end of table has been reached, if negative; whether the flight is an arrival flight, if zero; or if flight is a departure flight, if positive.

If the MH1 (I, 1) flag is negative, the program branches to statement number 835; assigns I to PH1, the number of the flight in MH1 last tested; and then branches to statement number 99999. If the MH1(I,1) flag is zero, then the flight is an arrival flight, and the program goes to the next flight listed in MH1. If the MH1 (I,1) flag is positive then the flight is a departure flight, and the program proceeds to statement number 833 which is the following statement. The program then sets ITEMP1 to MH1(I,6)*60, the time of flight in seconds from the simulation start. The program then tests if ITEMP1 is greater than save-value XFAXH, which is the maximum time interval between current time and flight time allowed for addition to the transfer flight table. This has a default value of 120 minutes. If ITEMP1 is greater than XFAXH, then the departing flight will leave after the maximum time interval and all departure flights after this departure will also leave after the time interval since the flights in MH1 are listed in chronological order. If ITEMP1 is greater than XFAXH, the program branches to statement number 835; assigns I to PH1, the number of the last flight tested in the flight table; and then branches to statement number 99999. If ITEMP1 is not greater than XFAXH, the program goes to the following statement.

The following statement tests whether ITEMP1 is less than savevalue XFDXH, which is the minimum time interval between current time and flight time allowed for addition

to the transit flight table. This has a default value of 30 minutes. If ITEMP1 is less than XFDXH, then the departure flight is scheduled to leave at too early a time to be added to the transfer flight table, so the program goes to test the next flight in the MH1 flight table. If ITEMP1 is not less than XFDXH the program goes to the following statement which tests whether MH1(I,11) is equal to zero or not. MH1(I,11) contains the number of transfer seats to be filled on the departure flight. If MH1(I,11) is zero, then there are no transfer seats to be filled, and the program goes to test the next flight in the MH1 flight table. If MH1(I,11) is greater than zero, the program goes to the following statement.

The following statement increments NOFXFR, which is the count of transfer flights in the transfer flight table, by 1. Next, the departure flight, I, is added to the transfer flight table by setting MH5(NOFXFR) to I. If all flights have been tested in the MH1 flight table, and flight time relative to simulation start does not exceed XFAXH, the program sets PH1 to I, the last row number in MH1 at statement 855; and then branches to the statement number 99999.

At statement number 832, the start of the DELETE FROM FLIGHT TABLE SECTION, the program first tests if MH5(1), the MH1 row number of the first flight in the transfer flight table, is not equal to IV2, the MH1 row number of the flight that is to be deleted. If the flight to be deleted is not the first flight listed in the transfer

flight table, then the program branches to statement number 99999. If they are the same flight, then the count of flights in the transfer flight table, NOFXFR, is decremented by one. The program then shifts each remaining flight in the transfer flight table one position toward the begining of MH5, thus deleting the first flight from the table. The program then branches to statement number 99999.

At statement number 830 which is the start of the add to transfer flight table section, the program first tests if NOFXFR, the count of flights in the table, is equal to 100. If it is, the program branches to statement number 831, writes the message; 'ADDITION OF DEPARTING FLIGHT, MH1 ROW NO', IV2 'TO TRANSFER FLIGHT TABLE MH5 WOULD HAVE CREATED OVERFLOW CONDITION. FLIGHT NOT ADDED', and then branches to statement number 99999. If NOFXFR is less than 100, then NOFXFR is incremented by 1 and MH5(NOFXFR) is set to IV2 which adds the flight to the transfer flight table. The program then branches to statement number 99999.

When transfer flights are deleted from the transfer flight table, GPSS fills unassigned transfer seats. The logic beginning with statement 836 obtains the point number of the airline ticket counter to initiate the processing of these transactions. At statement 836, the airline number of the flight is obtained from MH1(IV2,3) and assigned to IARLIN. The index number, IROW, for ticket facilities is obtained from MH8(14,2). The number of these facilities, INUMTC, is assigned from MH8(14,1). ITEMP1 is the row number of the first facility

in MH9 of this type and is set equal to IROWNO+1. ITEMP2 is the row number of the last ticketing facility in MH9 and is IROWNO+ INUMTC. Matrix MH9(I,4) is searched between the I subscript levels ITEMP1 and ITEMP2 for the airline number identical to IARLIN. When this is found the program branches to statement 838.

If no airline is found, the program sets I to ITEMP1, MH9(I,4) to ITEMP2, and then writes error messages and continues to statement 838.

At statement 838 the point number IPTNO is obtained from MH9(I,3). This is assigned to PH2. The program branches to 99999 and returns to GPSS.

## 4.18 MISCELLANEOUS GPSS ERROR CONDITIONS SECTION

This section is called from GPSS to record a variety of error conditions. The calling transactions are found on user chain ERROR at the end of the simulation run. The variable IV2 is set to IVALUE(2), the type of error. The program then branches to the section of the program for the type of error specified in IV2.

At statement number 901, the message, 'VEHICLE XAC', IVALUE(3), 'UNABLE TO MATCH WITH PAX AT DEPLANING CURB. CHECK USER CHAIN "ERROR" FOR THIS XAC', is written and then the program branches to statement 99999.

At statement number 902, the message, 'PAX XAC WITH GROUND TRANSPORT MODE', IVALUE(3), 'ENTERED BLOCK DPLCO. CHECK USER CHAIN "ERROR" FOR XAC NO', IV5, is printed, and then the program branches to statement number 99999.

At statement number 903 through 910 the statement is a CONTINUE. This is done so that more error messages can be easily added at a later time. The program then branches to statement number 99999.

4.19 FORMATTED REPORT SECTION

This section is called once when the time of end of simulation event has occurred. The variable Cl is set to IVALUE(2), which is the relative clock time. The rest of this section is repeated for each type of facility I, where I assumes the values 1 through 20.

The flag, NSWTCH, for undefined numbers of agents, is reset to zero. K is set to MH8(I,1), the count of facilities of type I. If K is zero, which indicates that there are no facilities of this type, the program branches to statement number 450, and the next facility type will be processed. J is next set to MH8(I,2), the index number of facility type I. K is set to K+J which is the row number of the last facility of type I in MH9. J is incremented by one to set it to the row number of the first facility of type I in MH9. If the facility type is gate, custom, security, check-in, ticketing, car rental, or immigration the program branches to statement number 400 and prints the title of the simulation, if there is a simulation title. If the facility type is not one of the above facility types, the simulation branches to statement number 450 where I is incremented by 1 and then the next facility type is processed. The program then branches according to facility type to a write statement which will print out the title for that facility report. After each write statement, the program branches to statement number 430 where the column headings for the facility report are printed out. The count of the number

of lines printed on the page, NCOUNT, is set to 11+NTLINS where NTLINS is the number of lines in the simulation title, and the number 11 takes into account the number of lines for the individual facility report title and the column headings.

The variable ITEMP1, is next set to FACQSX(I), which is the base value of the queue and storages for that facility type. The basic equation for calculating the subscript for queue or storage attributes is J=K*(N-1)+L where J is the subscript, N is the number of the facility in that type, and K and L are indexing constants. IQUER is set to 4*(ITEMP-1) which is part of the subscript for the queue attribute cumulative time integral. The indexing constant L will be added at a later time. IQUEI is set to IQUER+IQUER which is part of the subscript for some of the queue attributes. The indexing constant L, which will indicate which attribute is wanted, will be added at a later time. ISTOX is then set to 11*(ITEMP1)-1 which is part of the subscript for one of the storage attributes. ITEMP1 is then set to ITEMP1-FACQSX(I)+1 which sets the value of ITEMP1 to 1.

The segment of the program through statement number 455 is then repeated for each facility in type I, where N is incrementally set to MH9, row number J through K. The program first tests if the facility is a dummy facility by determining if MH9(N,3) is zero. If it is zero, the program branches to statement number 448. If MH9(N,3) is not zero, NCOUNT is then incremented by 2, to add to the count of lines printed

the number of lines needed to print the current line. If NCOUNT is less than or equal to 55, then a full page has not been printed yet, and the program branches to statement number 445. If NCOUNT is greater than 55, then a full page has been printed, and the program prints the message, "ALL TIMES IN MINUTES: SECONDS." The title of the simulation, if there is a title, is printed at the top of the next page. The program then branches to a write statement which prints out the title of the facility report at the top of the next page. After each write statement the program branches to statement number 443 where NCOUNT is set to 11+NTLINS to account for the number of lines used in the title and column headings. The column headings for the report are then printed. At the next statement, statement number 445, ITEMP2 is set to the current contents of storage plus number of available units in storage which gives the total number of agents in the storage facility. If ITEMP2 is greater than 1000, (1000 being an arbitrarily large number) then the number of agents in the storage is undefined, and the flag NSWTCH is set to 1. ITEMP3 is next set to the storage entry count times the scale to obtain the total number of patrons using the facility. If ITEMP3 is greater than zero indicating that the storage has been used, then the program branches to statement number 444. If ITEMP3 is not greater than zero, then the variables ITEMP4,XTEMP5, ITMP6M, ITMP6S are set to zero and the program branches to statement number 446. This is done in order to avoid division by zero and to avoid needless calculations. At statement

number 444, ITEMP4 is set to the maximum storage contents to obtain the maximum number of agents busy. ITEMP5 is set to the cumulative time integral divided by C1, the relative clock time, to obtain the average number of agents busy. ITEMP6 is set to the cumulative time integral divided by the entry count times the scale to obtain the average time per patron in seconds. ITEMP6M is set to ITEMP6/60 to obtain the seconds part of the average time per patron. At the following statement, which is at statement number 446, ITEMP7 is set equal to the total entry count times the scale.

If ITEMP7 is greater than zero, indicating that there were entries to the queue, then the program branches to statement number 447. If ITEMP7 is equal to zero, indicating that there have been no entires to the queue, then the variables ITEMP8, XTEMP9, ITM10M, ITM10S are set to zero, and the program branches to statement number 449. This is done to avoid dividing by zero and to avoid needless calculations.

At the following statement, which is at statement number 447, ITEMP8 is set to the maximum contents of the queue times the scale; XTEMP9 is set to the cumulative time integral for the queue times the scale divided by the relative clock time to obtain the average queue size. ITEMP10 is set to the cumulative time integral for the queue times the scale divided by the total entry count to obtain the average time in the queue in seconds. ITM10M is set to ITEMP10 divided by 60 to obtain the average time in the queue in integer minutes. ITM10S

is set to the remainder of ITEMP10 divided by 60 to obtain the seconds part of the average time in the queue. The data for the facility report is next written out.

At the following statement, which is at statement number 448, ITEMP1 is incremented by 1 to obtain the number of the next facility type I. IQUER is incremented by 4, IQUEI is incremented by 8, and ISTOX is incremented by 11 to obtain the subscripts for the next facility in type I. The following statement, which is at statement number 455, is a continue statement and is the last statement of the DO LOOP which prints the facility report for all facilities of type I.

The program then writes the message, 'ALL TIMES IN MINUTES: SECONDS'. If the undefined agent switch, NSWTCH, is set to 1, then the following message is written: '**INDICATES UNDEFINED(UNLIMITED) NO. OF AGENTS'. The following statement, which is at statement number 450, is a continue statement and is the last statement of the DO LOOP which processes all facility types 1 through 20. The program then branches to statement number 99999.

## 4.20 CLOCK UPDATE SECTION

This section is called once every minute of simulation time. ITEMP1 is set to the halfword save-value CLKXH plus IVALUE(2)/60 to obtain the new clock time. IVALUE(2) is the time increment in seconds which has been set to 60 in the GPSS program, and CLKXH is the clock time which is to be incremented. Since the clock time is kept in the form of hours and minutes, the program next determines if an hour has passed, by dividing ITEMP1 by 100 and checking the remainder to see if it is greater than or equal to 60. If the remainder is greater than or equal to 60, then an hour has passed and an hour is added in the clock column to the clock time by adding 40 to ITEMP1. The halfword savevalue CLKXH is then set to the new clock time, ITEMP1. The program then branches to statement number 99999.

## 4.21 SNAPSHOTS

This section produces two output time series. The first is the occupancy or congestion counts at simulated terminal points for each five-minute time interval. The output data, written on File 12, consists of the simulated time and number of persons currently located at this point. The second time series are flow and queue length data for selected simulated landside processors produced as a function of time. This data is written on Files 13 and 14.

The program stores the current clock time in ITEMP1, then tests LINSNP, the line counter for occupancy data for a value less than 50. When this condition occurs the program branches

to statement 653. When LINSNP is 50 or greater the program proceeds to the next instruction. LINSNP is made equal to NTLINS, the number runtitle records input for use as the simulation title. The title and the heading "5 MINUTE SNAPSHOTS OF CONGESTION AT POINTS" are written on File 12, with column headings for time and point numbers. Because the initial value of LINSNP is 50, this information is produced on the initial call to this section.

At 653, the halfword savevalues 1 to 24 of the GPSS MAIN program, representing simulated congestion at the corresponding point numbers, are stored in the ITEMPA array by a DO loop ending at statement 654. The ITEMP1 and ITEMPA values are written to File number 12, then LINSNP is incremented by one.

The remainder of the snapshot section produces the flow, queue length, and halfword savevalues for the corresponding GPSS entities with numbers specified by the GPSTO, GPQUE and GPHALF arrays discussed in the input section. A title is written for this information on File 13 using logic similar to that for congestion. The counter LINSNX is used as a line counter in place of LINSNP and is also initialized to 50. At statement 960 LINSNX is incremented by one.

A DO loop ending at 660 extracts the required entry counts, current contents, queue contents and savevalues to produce the time series. The GPSS storage number, ISTRNO, identifying the simulated landside processor for which flow data is to be extracted, is obtained from the input GPSTO(IR). When a storage number is not present in GPSTO(IR), the value is

zero for the element and the program branches to statement 965. When a storage number is provided, the subscripts JENTCT and JRCON are calculated by the following algorithm:

$$J = K*(N-1)+L,$$

where: J = Subscript value for GPSS addressing JENTCT, JRCON

K,L = Indexing contants

N = Index number of specific entity type ISTRNO

This formula is obtained from the IBM General Purpose Simulation System V User's manual (SH20-0851-1) pp. 164-167. The constants are provided by Table 12-1 of the referenced document. The cumulative entry count, XENTCT, and current contents, XRCON, are then obtained from ISTO (JENTCT) and ISTO (JRCON), respectively.

The variable flow, the number of passengers or vehicles processed by the storage in a specified time interval is the difference between the cumulative entry count, XENTCT, at the current clock time and the cumulative entry count, ENTRCT(IR), for the previous interval minus the change in current contents, XRCON-CRCON(IR), over the same time duration. The entire quantity is multiplied by the simulation scale factor SCALE.

After flow is calulated, the current cumulative entry count and contents are stored in ENTRCT(IR) and CRCON(IR), respectively, for use in the succeeding time interval calculation. The initial values in these arrays are zeroes.

The output array element, TSSOUT(1), is assigned the value ITEMP1 and TSSOUT(IR+1) is made equal to FLOW.

This queue length present at a landside processor is obtained next from the GPSS MAIN program. The number of the designated queue, ITQUE, is obtained from the input GPQUE(IR) at statement 965 and tested for zero in the next statement. When the element is zero, the program branches to 967. The subscript JQUE is calculated from the same algorithm as JENTCT and JRCON. Current contents of the queue are obtained from IQUE(JQUE). These are multiplied by the scale factor and stored in TSQUE(IR+1). The current time is stored in TSQUE(1).

At statement 976, the GPSS halfword savevalue designated by GPHALF(IR) is stored in ITHLF. Again, as in the flow and queue length subsections, the value of ITHLF is tested for zero and the program branches to 660 for this condition. Because the only information that GPSS stores for the Halfword savevalue is the current value of the savevalue, no calculation is required for the subscript. The value is directly obtainable as ISAVEH(ITHLF) and assigned to ISHLF.

Halfword savevalues are generally used to record cumulative processor outflows in the GPSS program when storage entry counts and current contents are inapplicable. The value FLOW is calculated by subtracting, ISHLF, the current magnitude of the savevalue from JTHLF(IR), the value from the last time interval. This difference is multiplied by SCALE. The current value of FLOW is stored in TSHALF(IR+1) and the clock time,

ITEMPl, is stored in TSHALF(1). The current value of ISHLF is stored in JTHLF(IR) for use in the next time interval. The initial value of JTHLF(IR) is zero.

The IR loop ends at 660 with a CONTINUE statement.

A DO loop ending at 969 calculates outflow from security stations and stores them in TSFLOW(IL+1). The security outflow is recorded in the GPSS MAIN program in halfword matrix 12. The cumulative flow value, JSECFL, is obtained from HMH12 and the flow during the current time interval is calculated with the same procedure used for savevalues. At 969 the current security flow is stored in ISECFL(IL). Initial values of ISECFL are also zero.

The outflow of simulated full service airline counters are recorded in MH13 and stored in TTFLOW by a DO loop ending in 923. Processing is identical to security flow calculation and storage.

The values stored in TSSOUT, TSQUE, TSHALF, TSFLOW and TTFLOW are written on File 13 for print out. These are also written as a single record on File 14 under a 100I5 format for later processing and averaging with other ALSIM runs. The section ends with a GO to 99999 instruction to return to the GPSS MAIN program.

## 4.22 CHANGE CARD PROCESSING

This section provides a method for changing numbers of servers at landside facilities as a function of time. Data cards specifying time, facility name and numbers of servers must be input. This section is called from GPSS whenever a change is required.

The argument IVALUE(2) is tested for a value of 2, the flag signifying a decrease in the number of servers in a storage. A value of 2 causes branching to 590 to accomplish this. The other value, 1, is used to read and process a change card.

The variable ICHNG1 is tested for the initial value of zero. If true, the program branches to statement 580. to read the initial change card and return to GPSS. Otherwise the variable SERVERS(1), which contains the characters representing any facility, is tested for zero. If a zero is found, indicating no data present on the input card, the program branches to statement 560 for reading the next change card. For non-zero values of SERVRS(1), a search through the facility type array, FACTYP, is performed at statement 551. Variables I and M are initialized to 1 and 0, respectively. SERVRS(I) is compared with FACTYP(L) in a DO LOOP, with L ranging from 1 to 20. When the characters match, the program branches to 553. If no match is found, the program branches to 557 to write an error message and terminate the program.

At statement 553 the GPSS storage number FACQSX(L) is assigned to J. The value of J is tested for zero. When J equals zero, the facility is not defined in the simulation and the program branches to 557. For non-zero values of J, a value of one is subtracted from J and I is incremented by 1. The next item on the data card, SERVRS(1), represents the facility number within type and is assigned to IFACNO. If IFACNO is zero, indicating the end of the data stream, the program branches to 558. At 558 the array SERVRS(I) is zeroed. The number of changes, M, is placed in the savevalue NSCXH, and the program continues to statement 560.

When IFACNO is less than zero, a new facility name is present in SERVRS(I) and the program branches back to statement 551 to process the next facility type. If IFACNO is greater than the number of facilities within type, NFASCM (L,1), an error is recognized and the program branches to 557.

When IFACNO is an admissible value, the subscript, K1, used to obtain current contents of the storage from ISTO(K1) is calculated using 11*(J+IFACNO-1) +1. The subscript K2 is K1+1 and provides the remaining storage capacity from ISTO(K2). Variables ICONT and IRCAP are set to current contents and remaining capacity, respectively.

The next value in SERVRS(I) is obtained by incrementing I by 1. This provides the new number of servers at the facility and is set to NEWCAP. If the value of NEWCAP is less than zero the program branches to 557. When NEWCAP is greater than or equal to the current contents, ICONT, the program

branches to 555. At 555 the remaining capacity, ISTO(K2), is changed to the value NEWCAP minus ICONT. The index M of MH7 is increased by 1 to point to the row number used for the MH storage on a change data card. The GPSS storage number given by J+IFACNO is stored at MH7(M,1). The count ISTO(K1+5) is decreased by 1 to compensate for the condition occuring when the new capacity is greater than or equal to the current contents and the storage is full. The GPSS program inserts a transaction into the storage under these conditions to allow transactions waiting on the delay chain to start moving. The program branches back to statement 554. When NEWCAP is less than the current contents, the remaining capacity ISTO(K2) is zeroed. The index M is increased by 1 and the matrix element MH7(M,1) is made equal to the GPSS storage number given by J+IFACNO. The element MH7(M+30,1) is made equal to the new capacity NEWCAP. The program branches back to 554 to process the remaining storages on the change data card.

The error condition occurring when an input facility number to be changed cannot be recognized by the model, causes branching to 557. At this location, an error statement specifying time of occurrence and other parameters is printed out. The logic switch JOBLS is set and the program returns to GPSS for immediate termination.

After processing a change card the program continues to statement 560 and then reads the next change card. The FORTRAN input data card is read into the ICARD array and the number of cards, NCARD, and line count, LINECT, are both incremented by 1.

LINECT is tested for a value of 51 to determine if a page is to be printed in full. If LINECT is less than 51 the data card is printed immediately. If LINECT equals or exceeds 51, a new page is started and the data card printed out. The program proceeds to statement 580, the location that the program branches to when this section is utilized initially. The first change card is assumed to follow all other landside simulation program data cards and is read in the DATA INPUT SECTION of the FORTRAN program. At statement 580, card identifiers are tested to determine identity with the variable ICHAN which contains the character string 'CHAN'. An incorrect card type causes branching to 585.

The flag ICHNG1, initially having a value of zero in order to cause branching to 580 for the first entry, is now set to one. Subroutine XCODE is called and an in-core write into the array, BUFFER, is performed on the card image. The first word of array, BUFFER, is set equal to NAMECH which is the character string '&CH' for an ensuing namelist read. The second word is modified to blank the fifth and sixth characters on the data card and preserve the seventh and eighth characters by a logical AND, plus the addition of the hexadecimal number in variable BLANK2. XCODE is again

called and a read statement is executed with the namelist of CH.

The variable IC is set to the simulation clock time CLKH. The time interval in seconds from current simulation time, IC, until the next change occurring at the time indicated by the variable, TIME, is calculated and placed in fullword savevalue CHGXF. The program returns to GPSS.

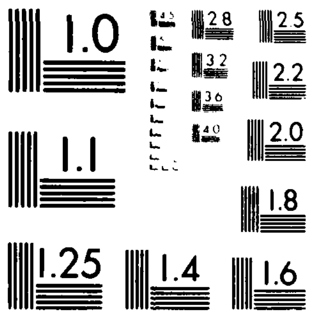
Data cards not recognized as change cards or an end of input file cause branching to 585. The program makes CHGXF equal to $10^6$ indicating no further changes and returns to GPSS.

When the storage capacity must be lowered, that is , the number of servers decreased, the initial statement of this FORTRAN program section caused branching to statement 590. At this location the subscript J for current contents of the storage number contained in IVALUE(3), obtained by GPSS from MH7 (M, 1), is calculated. The new capacity, INVALUE(4), obtained in GPSS from MH7(M+30,1), is placed in NEWCAP. The difference, NURCAP , between new capacity, NEWCAP, and current contents, ISTO(J), is calculated and tested for a value greater than or equal to zero. If true, the new capacity equals or exceeds the current contents and the program branches to 592. At 592 the remaining capacity ISTO(J+1) is made equal to NURCAP . The flag, SCLXH , is given a value of one to indicate that the storage capacity lowering process is complete. The program returns to GPSS.

When current contents exceed the new capacity, NURCAP is less than zero. The program makes the remaining capacity

AD-A117 603  TRANSPORTATION SYSTEMS CENTER CAMBRIDGE MA  F/G 1/5
AIRPORT LANDSIDE. VOLUME V. APPENDIX B. ALSIM SUBROUTINES.(U)
JUN 82  L MCCABE, M GORSTEIN
UNCLASSIFIED  DOT-TSC-FAA-82-4-5  FAA-EM-80-8-5  NL

2 of 4
AD
A117603


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

ISTO(J+1) equal to zero and returns to GPSS to wait until a transaction leaves the storage and this section is again accessed.

4.23 CONCESSION SECTION

This section is called by transfer passengers who are waiting in the terminal before catching their connecting flight. The value of NOCONC is first tested for a value of zero. If NOCONC is zero then there are no concessions defined by the input data, and the program branches to 752. If NOCONC is not equal to zero then the variable NPTFM is set to IVALUE(2), the current location. The variable IFLT is set to IVALUE(3), the flight table row number. The variable IGAT is set to MH1 (IFLT,9), the gate number for flight IFLT. The variable I is set to zero, which indicates that the concession is in the lobby. If IVALUE(6) is equal to 2, the flag that the concession to be found is in the concourse, then I is set to MH9(IGAT,4), the number of the security for gate IGAT. The variable L is set to INDEXF (15) +1, the subscript for the first concession facility in the MH9 facility matrix. The variable M is set to INDEXF(15) +NOCONC, the subscript for the last concession facility in the MH9 facility matrix. The variable IC, which will be used as a count of the concessions found with the correct location, is set to zero. The concession facilities in the MH9 facility are then searched through for an associated security whose number is the same as I. For each such security found IC is incremented by one. If the concession wanted is a lobby concession then I is zero and each concession with an MH9(J,4) value of zero is also a lobby concession and IC is incremented by one for each such case.

Next, IC is tested for a value greater than zero. If it is greater than zero, indicating concessions were found in the right location, then the program branches to 753. If it is not greater than zero, then at statement number 752 a zero is assigned to halfword parameter 5, and savevalue TRVXH is set to zero which give a zero waiting time and zero travel time to concession, respectively. The program then branches to 99999.

At statement 753, the variable IRN is set to the remainder of IVALUE(4) , which is a random number between 1 and 999, divided by IC plus one. The result is the number of the concession chosen in a random manner. IC is set to zero and the MH9 facility matrix is searched again for concessions which have an associated security which is equal to I, or which are lobby concessions if I is zero. For each such concession found, IC is incremented by one. When IC is equal to IRN, the chosen concession number, the program branches to statement number 755.

At statement number 755, the variable NPTTO is set to MH9(J,3),the point number of the chosen concession. The statement number 756 is assigned to NEXT, and the program branches to statement number 950 to determine the walking time.

After the walking time is determined, at statement 756, the variable IC1 is set to IVALUE(5), the current clock time. The variable ITIM is set to MH1(IFLT,6)*50-IC1, the time remaining in seconds before the flight departs. If IVALUE(6) is equal to 1, indicating a lobby concession, then ITM is set

to ITIM-LEAVEL-LEAVEL*IVALUE(4)/1000, where LEAVEL is the latest time before flight time to leave the concession. LEAVEV is the spread of the uniform distribution before the latest time that the passenger will leave the concession. LEAVEV is multiplied by the random numbers IVALUE(4)/1000 which gives a random value between 0 and 1. The value in ITIM, as a result of this statement, is thus the amount of time the passenger will spend at the concession. If IVALUE(6) equals 2, indicating a concourse concession, then ITIM is set to ITIM-LEAVEC-LEAVEV*IVALUE(4)/1000, where LEAVEC is the latest time before flight time that the passenger will leave the concourse concession. If ITIM is less than zero, indicating there is not much time before the flight, then ITIM is set to zero.

Halfword parameter 2 is set to NPTTO, the point number of the concession. Halfword parameter 5 is set to ITIM. Halfword parameter 7 is set to J, the MH9 subscript of the concession. Byte parameter 11 is set to 15, the process code for concession. The program then branches to statement 99999.

## 4.24 CONCOURSE SECTION

This section is called each time a deplaning passenger leaves a concourse. NPTFM is set to IVALUE(2), the number of the point at which the passenger is coming from. IV3 is set to IVALUE(3), which is the gate number the passenger came from. ISEC is set to MH9(IV3,4), the security facility number (concourse) for gate number IV3. There are no actual concourse facilities in this simulation. The entrance and exit to a concourse are considered to be at the same place as the security facility, so that the number of the concourse and the point number for the concourse are taken to be the same as the facility number and the point number of the security at the concourse entrance, respectively. J is next set to INDEXF(3), the index number for security facilities, plus ISEC, to obtain the MH9 row number for security (concourse) number ISEC. NPTTO is then set to MH9(J,3), the point number for security (concourse) number ISEC. Statement number 920 is assigned to the variable NEXT and the program then branches to statement number 950 to determine the walking time.

After the walking time has been determined, the program branches back to statement number 920. NPTTO, the point number of the security (concourse), is assigned to halfword parameter 2. ISEC, the facility number of the security (concourse), is assigned to halfword parameter 5. The program then branches to statement number 99999.

## 4.25 WALKING TIME CALCULATION SECTION

This section is called from other parts of the FORTM program every time there is a need for a walking time determination. The flag NPTOSW is tested for a value of one. If it is equal to one, then a non-positive value of a point has been previously discovered. If NPTOSW is equal to one, then the program branches to 951 in order to skip the error message so that the error message will not repeat itself. If NPTOSW is not equal to one, then NPTFM and NPTTO, the point numbers that the transaction is going between, are tested for a greater than zero value. If both MPTFM and NPTTO are greater than zero then the program branches to 951. If either or both NPTFM and NPTTO are less than or equal to zero, then the point number or numbers are undefined, and the flag NPTOSW is set to one and an error message is written. At 951, halfword savevalue TRVXH is set to MH6(NPTFM,NPTTO) to obtain the walking time in seconds between the points. NPFTM is the number of the point the passenger is coming from, and NPTTO is the number of the point the passenger is going to. MH6 contains the walking time in seconds between all points in the airport configuration. ITEMPT is next set to halfword parameter PH9, which contains the cumulative walking time in seconds for that passenger, plus halfword savevalue TRVXH, to obtain the new cumulative walking time. The new cumulative walking time, ITEMPT, is then saved by assigning it to halfword parameter 9. The program then branches back to the section of the program that called it via an assigned GO TO statement.

## 4.26 ERROR ABEND AND END OF PROGRAM SECTION

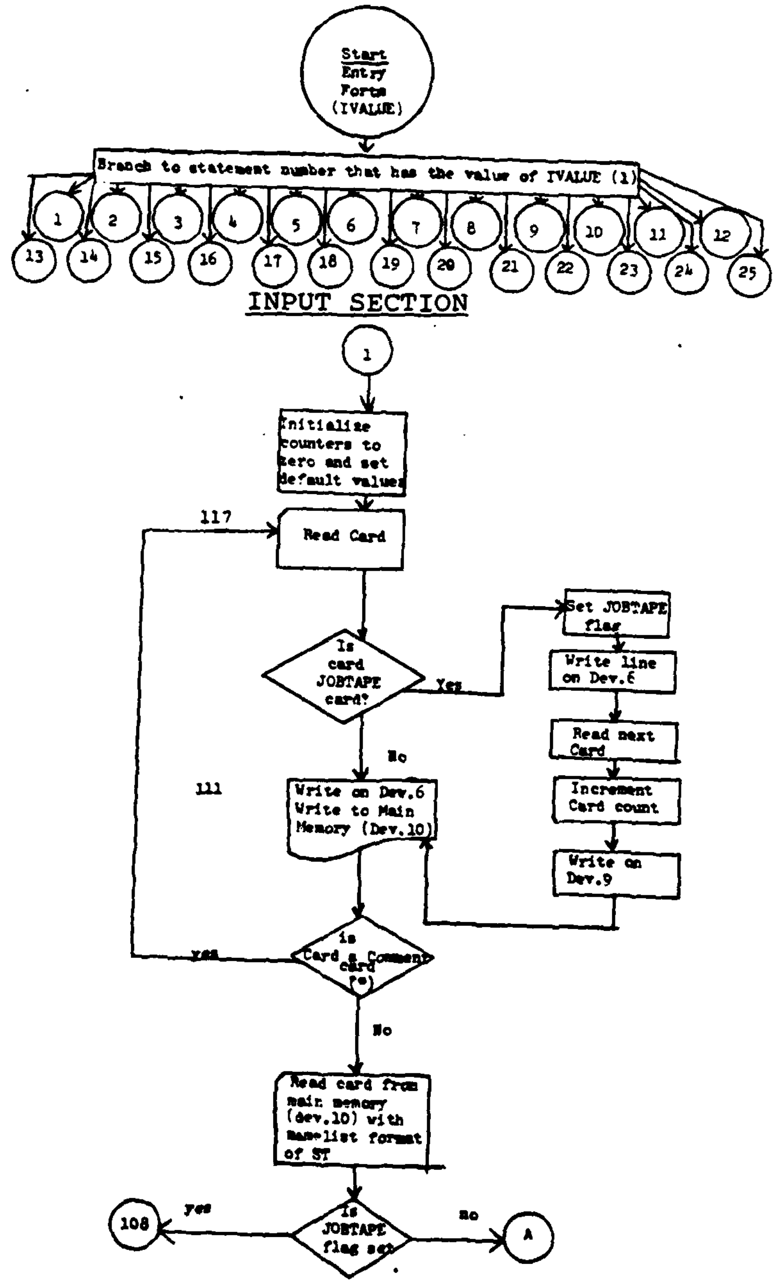
The ERROR ABEND SECTION is called from other parts of the FORTM program whenever the error count exceeds ERRORS, the maximum allowable number of errors. ERRORS has a default value of 50. The message, 'ERROR END - PROGRAM TERMINATING DUE TO ERROR COUNT EXCEEDING "ERROR"', is written; and logic switch JOBLS is placed in the set position. When control returns to the GPSS program from the FORTM program, this switch is always tested. When this switch is found to be in the set position, the simulation is halted. The program then branches to statement number 99999.

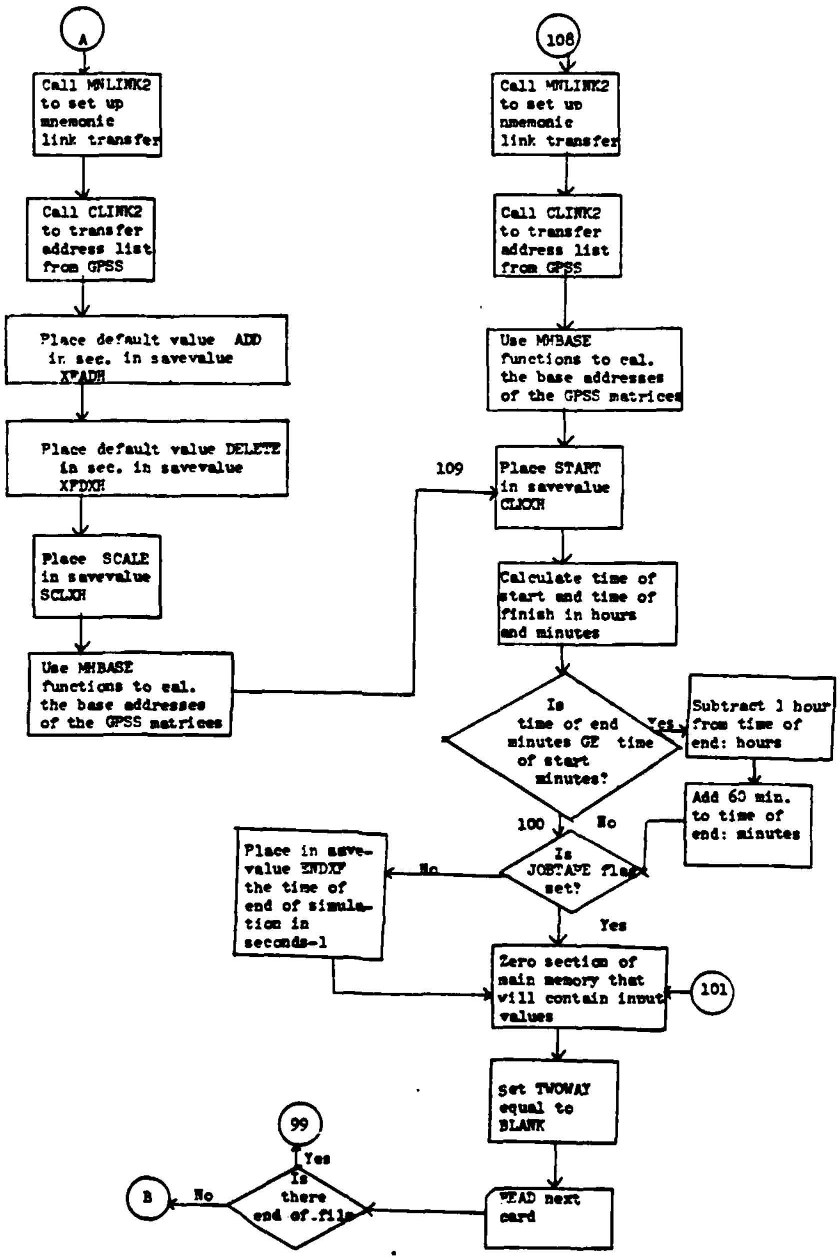
After the ERROR ABEND SECTION of the FORTM program there is a list of CONTINUE statements with statement numbers 1 to 25 which act as dummy sections. All of these statements are commented out due to the fact that there is an active section which has that statement number as its beginning point. If an active section is deleted then the corresponding CONTINUE statement should be uncommented in this section.

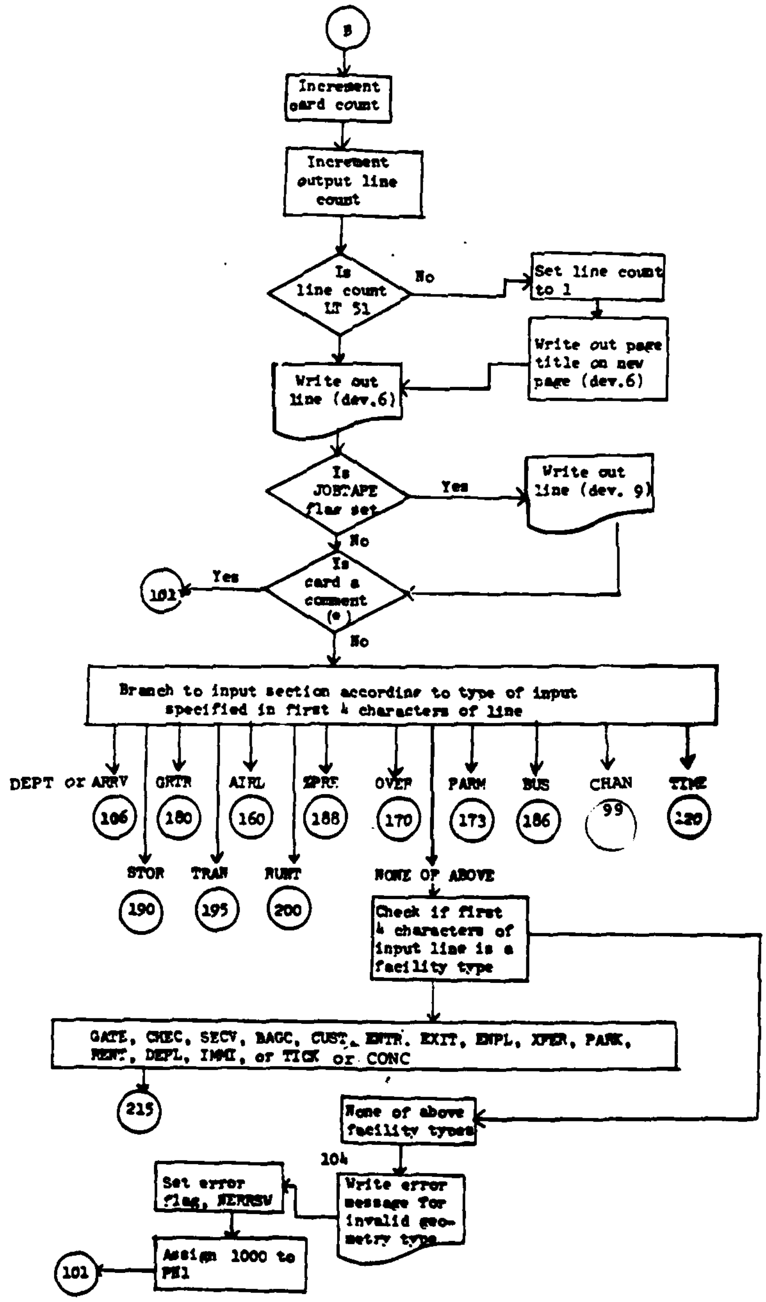
Statement 99999 is a RETURN. This is the only exit from the FORTM program back to the GPSS program. Finally, all the format statements for the FORTM program are listed at the end of the program.

# APPENDIX B-2

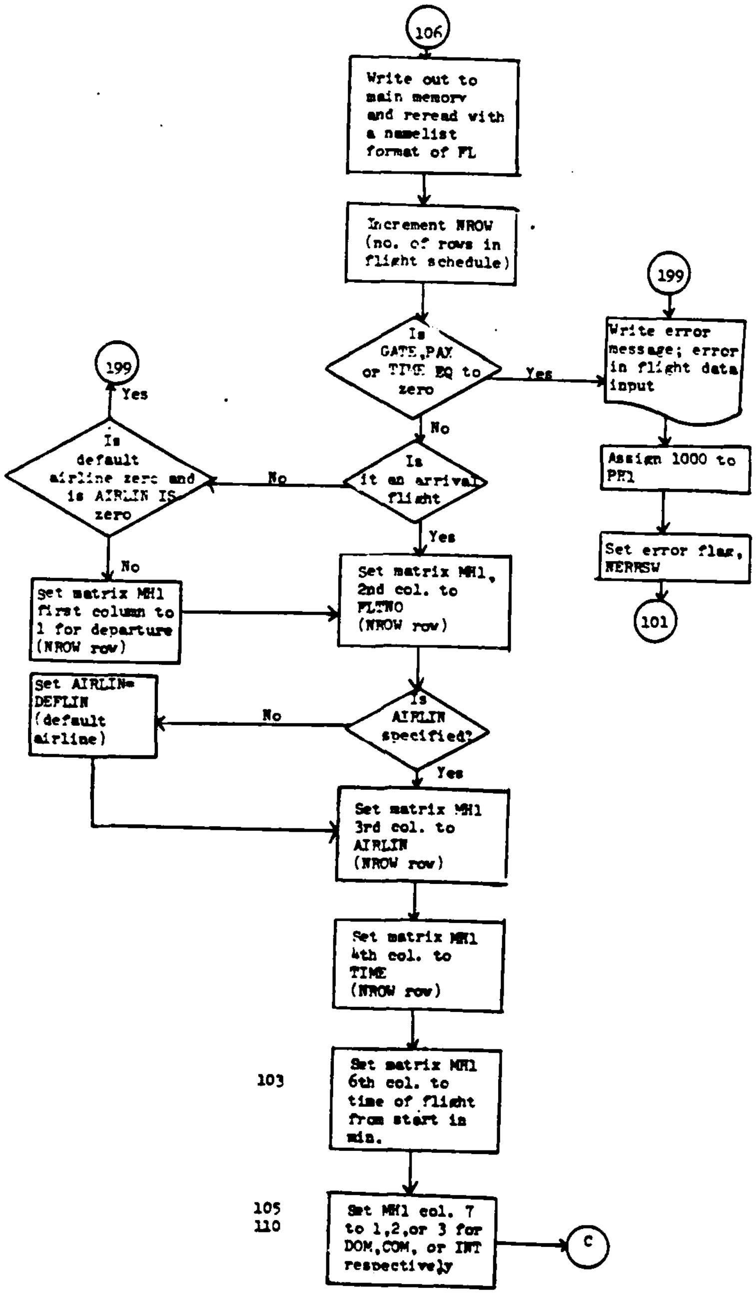
## FLOWCHARTS FOR FORTM SUBPROGRAMS

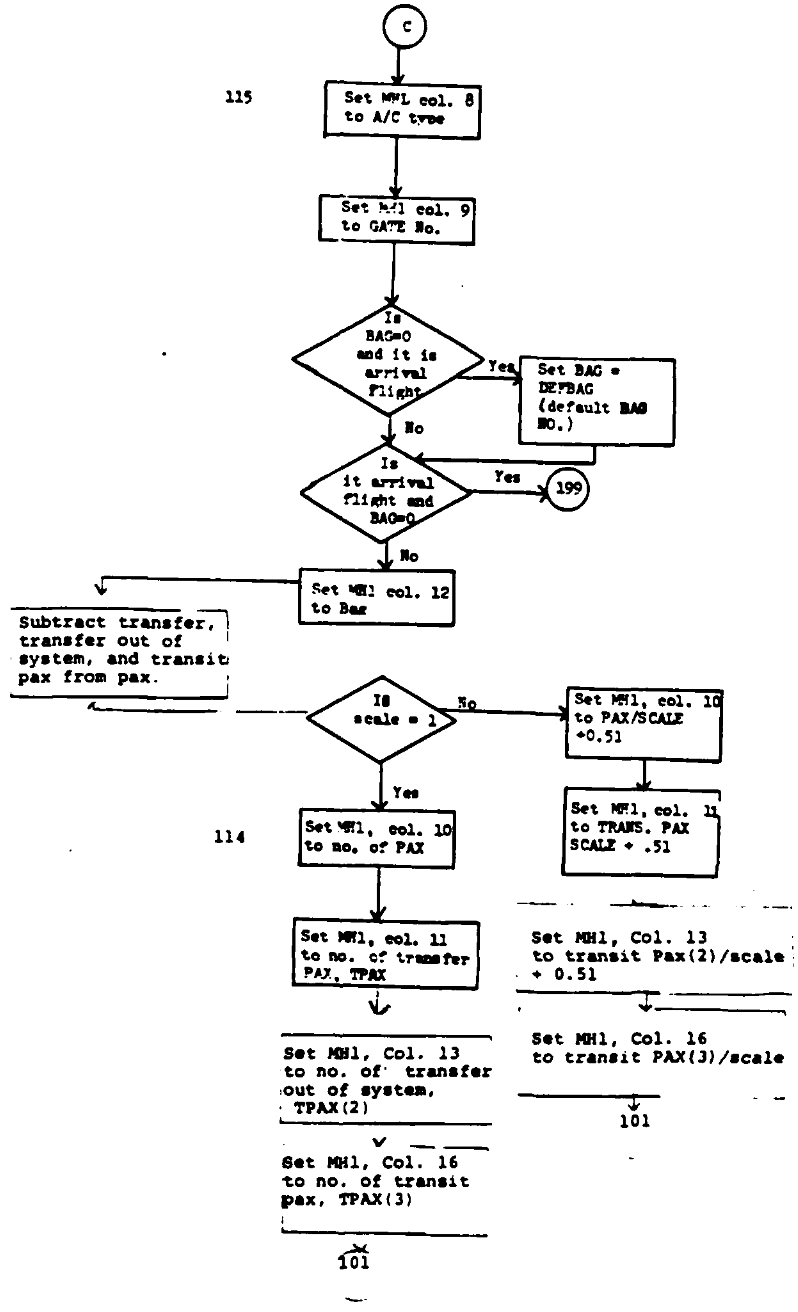

Start
Entry
Forts
(IVALUE)
Branch to statement number that has the value of IVALUE (1)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
INPUT SECTION
1
Initialize counters to zero and set default values
117
Read Card
Is card JOBTAPE card?
Yes
Set JOBTAPE flag
Write line on Dev.6
Read next Card
Increment Card count
Write on Dev.9
No
111
Write on Dev.6 Write to Main Memory (Dev.10)
Is Card a Comment card (*)
yes
No
Read card from main memory (dev.10) with namelist format of ST
Is JOBTAPE flag set
yes
no
108
A


A
Call MNLINK2 to set up mnemonic link transfer
Call CLINK2 to transfer address list from GPSS
Place default value ADD in sec. in savevalue XFADH
Place default value DELETE in sec. in savevalue XFDXH
Place SCALE in savevalue SCLXH
Use MHBASE functions to cal. the base addresses of the GPSS matrices
108
Call MNLINK2 to set up mnemonic link transfer
Call CLINK2 to transfer address list from GPSS
Use MHBASE functions to cal. the base addresses of the GPSS matrices
109
Place START in savevalue CLKXH
Calculate time of start and time of finish in hours and minutes
Is time of end minutes GE time of start minutes?
Yes
Subtract 1 hour from time of end: hours
Add 60 min. to time of end: minutes
100
No
Is JOBTAPE flag set?
No
Place in savevalue ENDXF the time of end of simulation in seconds-1
Yes
Zero section of main memory that will contain input values
101
Set TWOWAY equal to BLANK
READ next card
Is there end of file
Yes
99
No
B

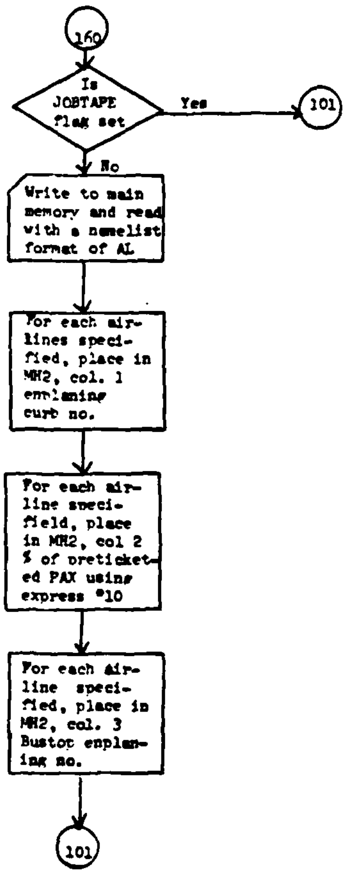

B
Increment card count
Increment output line count
Is line count LT 51
No
Set line count to 1
Write out page title on new page (dev.6)
Write out line (dev.6)
Is JOBTAPE flag set
Yes
Write out line (dev. 9)
No
Is card a comment (*)
Yes
101
No
Branch to input section according to type of input specified in first 4 characters of line
DEPT or ARRV
106
GRTR
180
AIRL
160
SPRE
188
OVER
170
PARM
173
BUS
186
CHAN
99
TIME
120
STOR
190
TRAN
195
RUNT
200
NONE OF ABOVE
Check if first 4 characters of input line is a facility type
GATE, CHEC, SECV, BAGC, CUST, ENTR, EXIT, EMPL, XFER, PARK, RENT, DEPL, IMMI, or TICK or CONC
215
None of above facility types
104
Write error message for invalid geometry type
Set error flag, NERRSW
Assign 1000 to PNI
101

# FLIGHT SCHEDULE INPUT

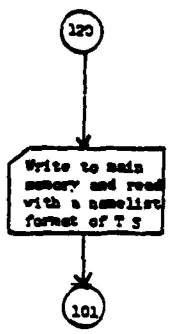



C
115
Set MHL col. 8
to A/C type
Set MHl col. 9
to GATE No.
Is
BAG=0
and it is
arrival
flight
Yes
Set BAG =
DEFBAG
(default BAG
NO.)
No
Is
it arrival
flight and
BAG=0
Yes
199
No
Set MHl col. 12
to Bag
Subtract transfer,
transfer out of
system, and transit
pax from pax.
Is
scale = 1
No
Set MHl, col. 10
to PAX/SCALE
+0.51
Set MHl, col. 11
to TRANS. PAX
SCALE + .51
Yes
114
Set MHl, col. 10
to no. of PAX
Set MHl, col. 11
to no. of transfer
PAX, TPAX
Set MHl, Col. 13
to no. of transfer
out of system,
TPAX(2)
Set MHl, Col. 16
to no. of transit
pax, TPAX(3)
101
Set MHl, Col. 13
to transit Pax(2)/scale
+ 0.51
Set MHl, Col. 16
to transit PAX(3)/scale
101

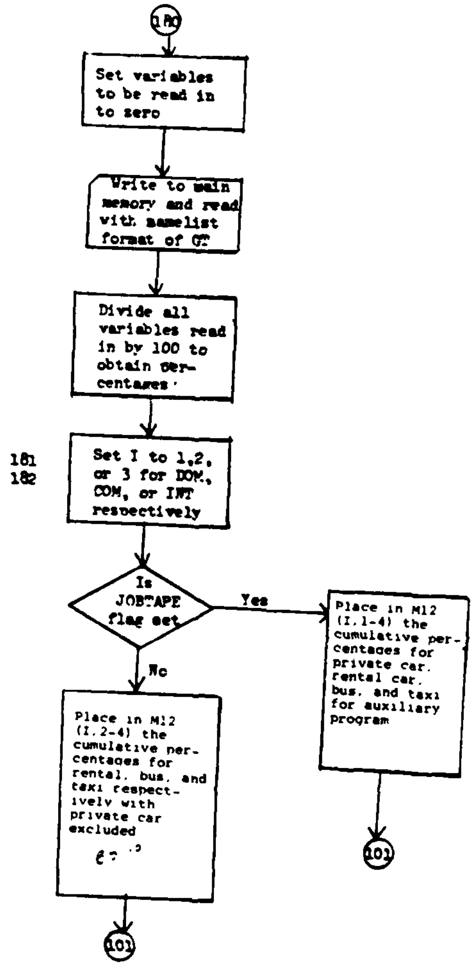
## AIRLINE DATA INPUT



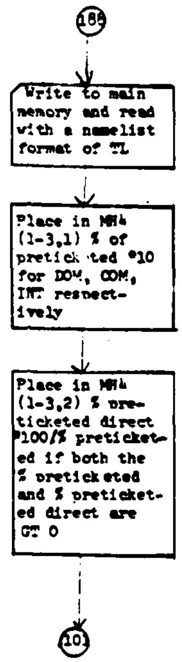

## TIME SERIES SPECIFICATIONS INPUT



## GROUND TRANSPORT INPUT

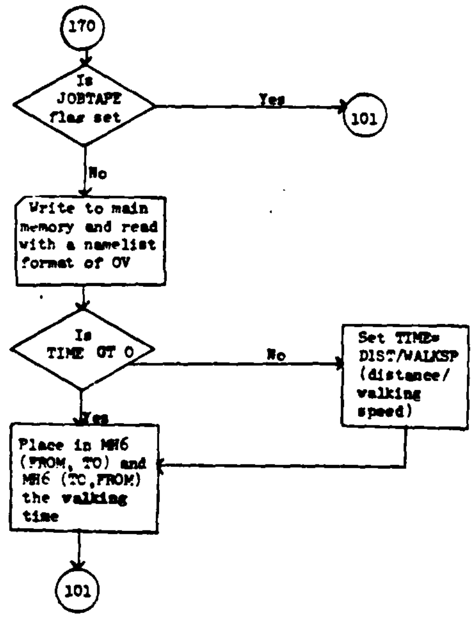


# PRETICKETED PAX INPUT

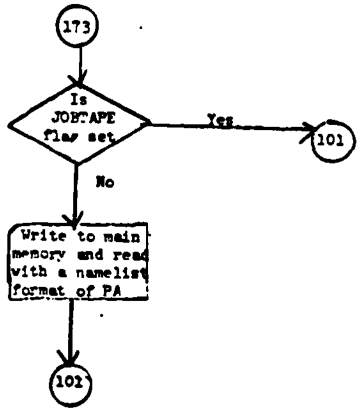


## WALKING TIME/DIST. OVERRIDE INPUT

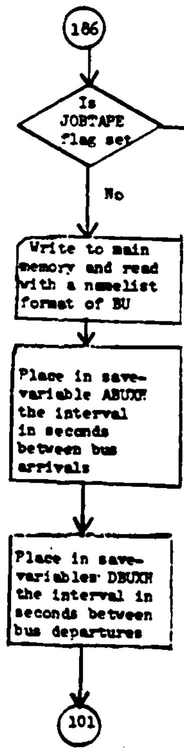
171



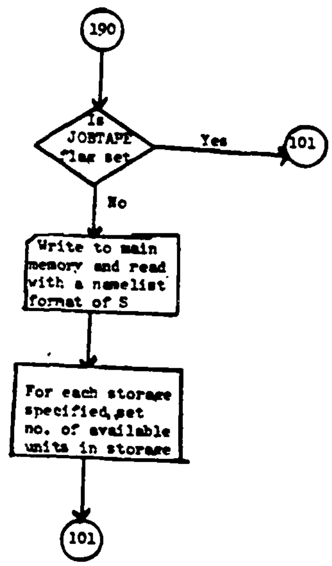
## PARM CARDS INPUT



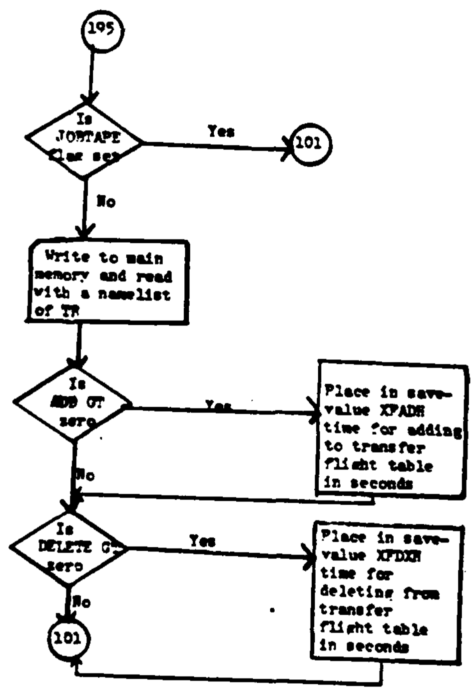

## BUS SCHEDULE INPUT



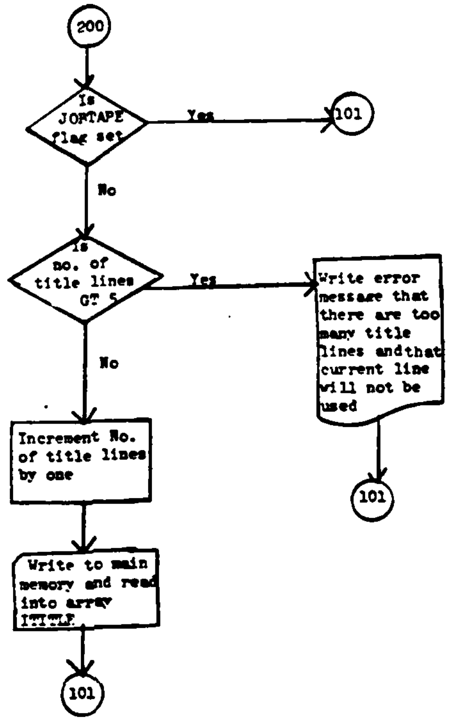
## GPSS STORAGE CAPACITY INPUT



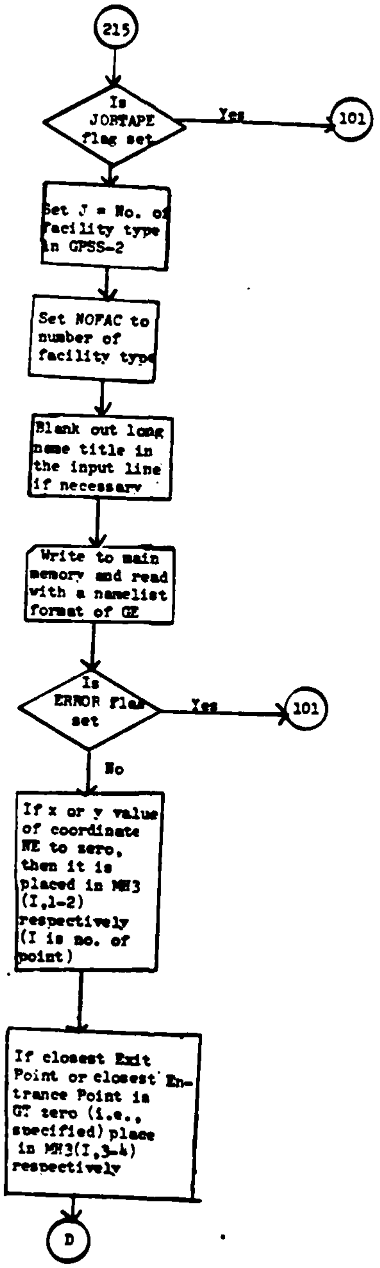

## TRANSFER FLIGHT OVERRIDE INPUT

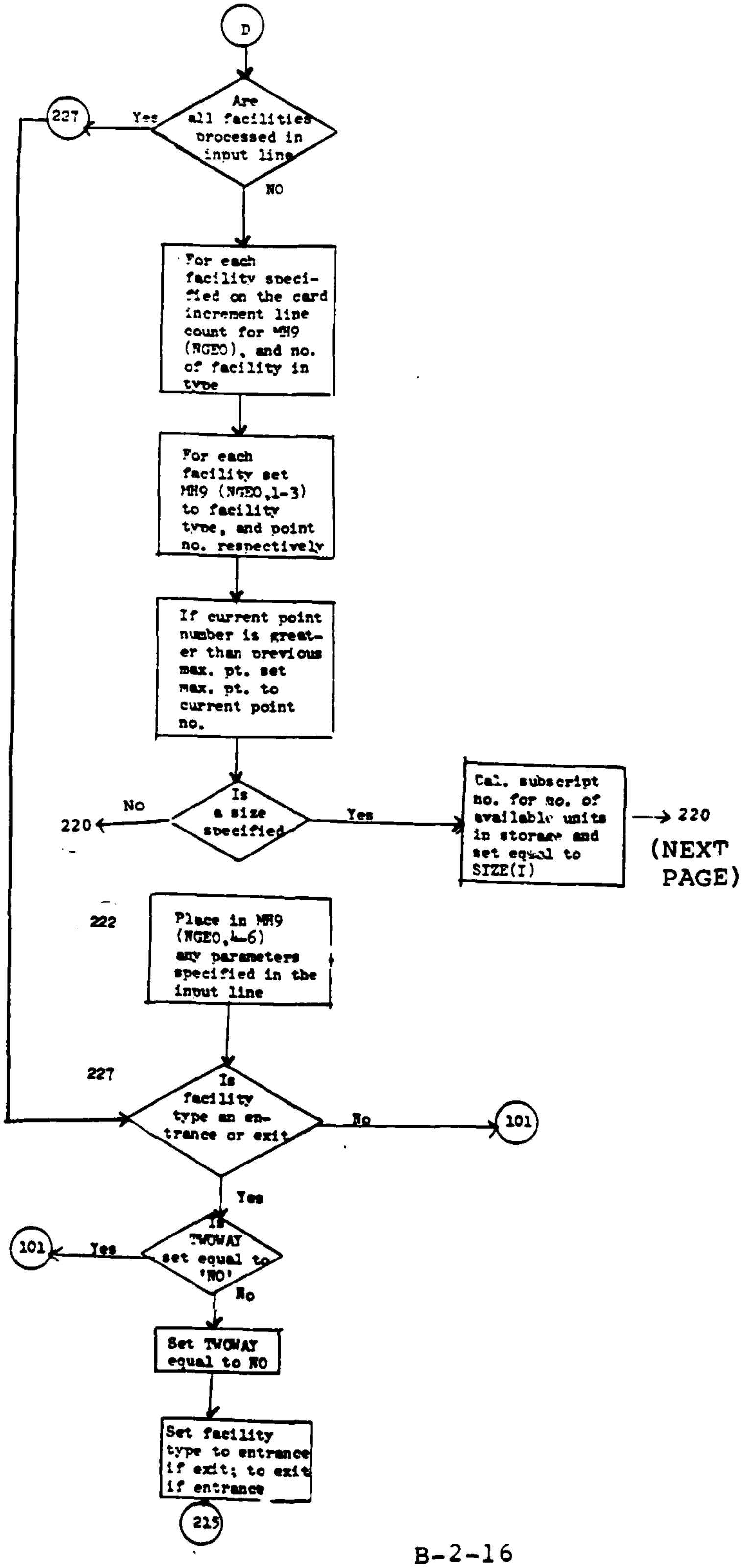


RUNTITLE CARD INPUT



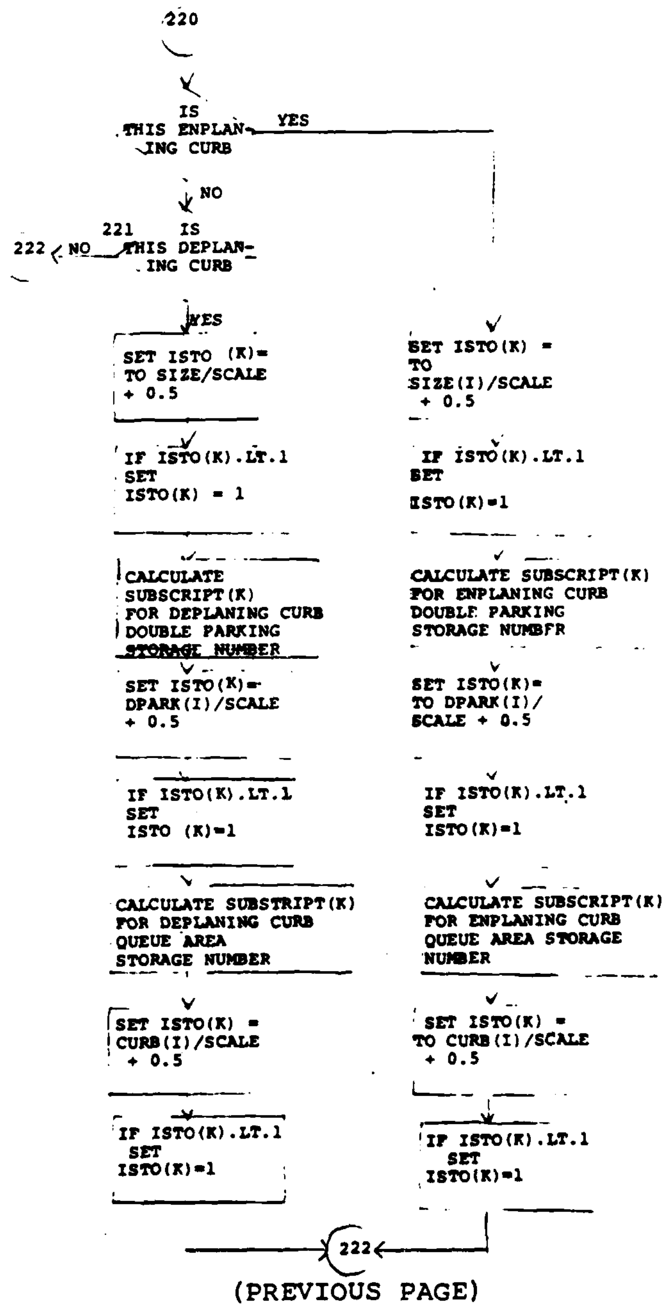
## GEOMETRY INPUT



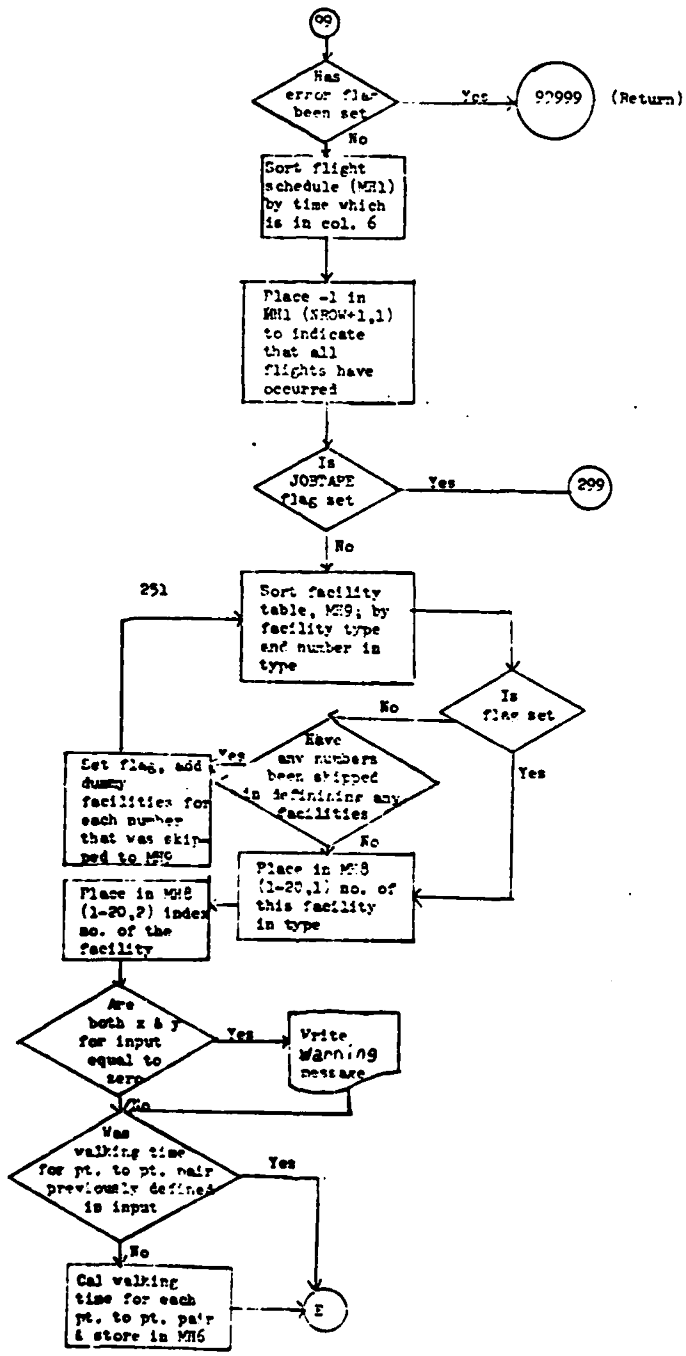


D
Are all facilities processed in input line
Yes
227
NO
For each facility specified on the card increment line count for MH9 (NGEO), and no. of facility in type
For each facility set MH9 (NGEO,1-3) to facility type, and point no. respectively
If current point number is greater than previous max. pt. set max. pt. to current point no.
Is a size specified
No
220
Yes
Cal. subscript no. for no. of available units in storage and set equal to SIZE(I)
220
(NEXT PAGE)
222
Place in MH9 (NGEO,4-6) any parameters specified in the input line
227
Is facility type an entrance or exit
No
101
Yes
Is TWOWAY set equal to 'NO'
Yes
101
No
Set TWOWAY equal to NO
Set facility type to entrance if exit; to exit if entrance
215

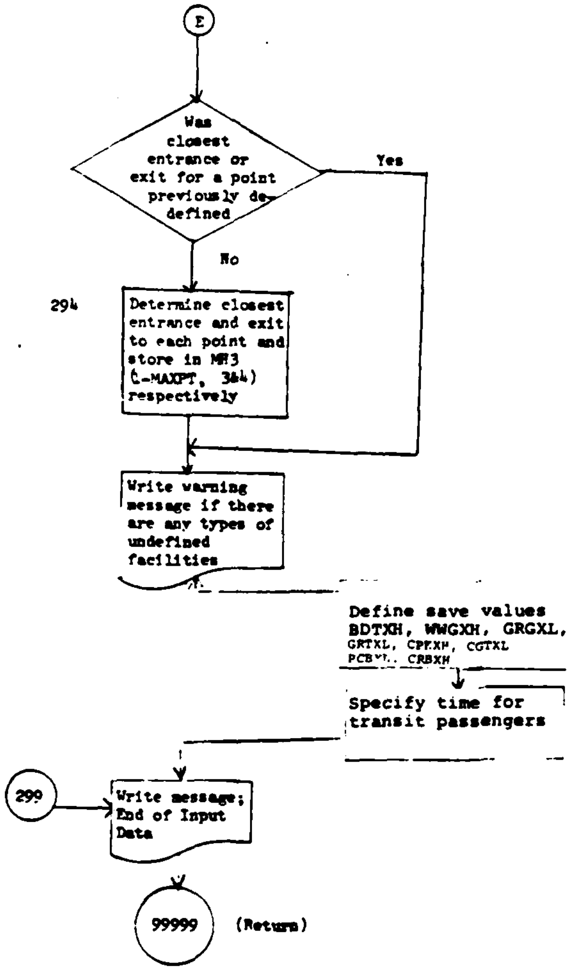
ENPLANING AND DEPLANING CURB STORAGE ASSIGNMENT



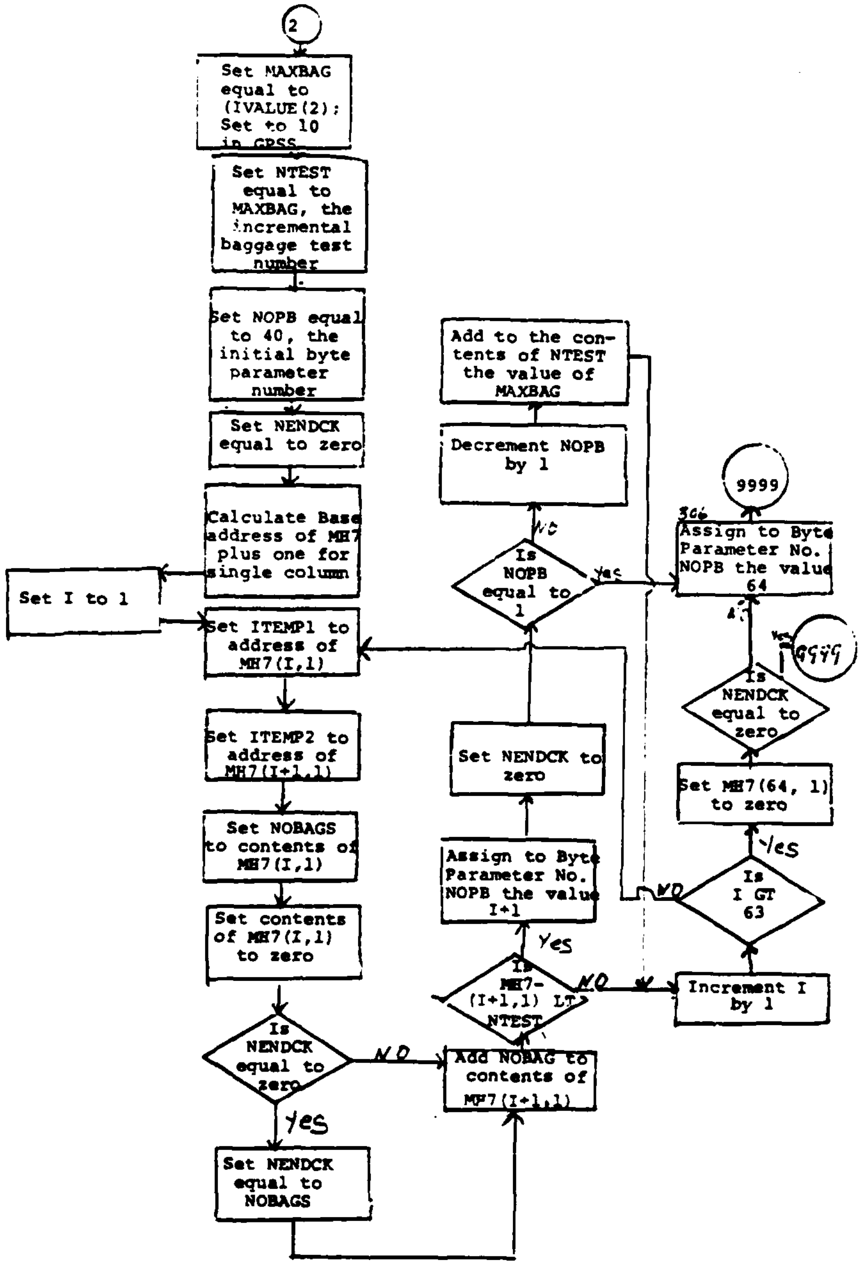
## FLIGHT SCHEDULE AND FACILITY SORT; WALKING TIME CALCULATION



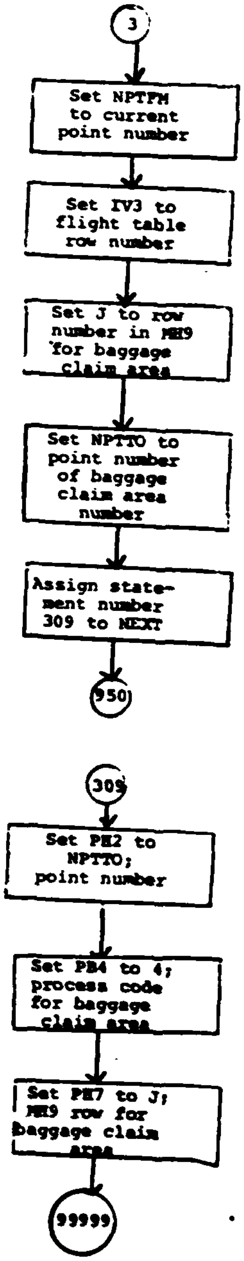
<u>CLOSEST ENTRANCE AND EXIT; END OF INPUT</u>



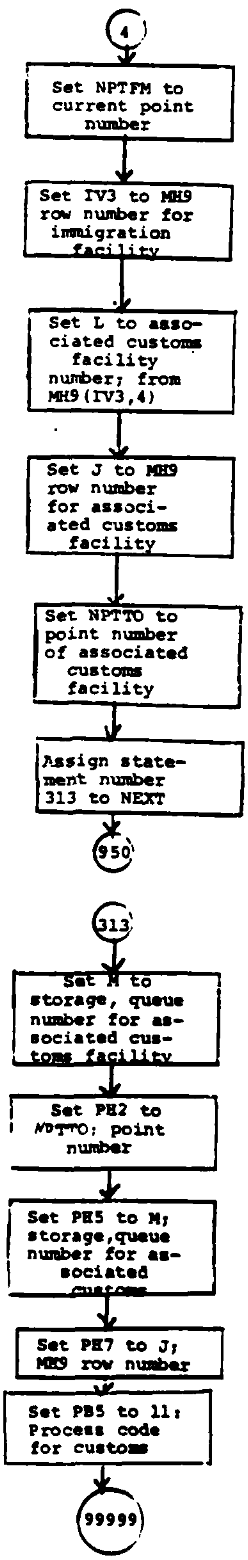
BAGGAGE UNLOAD



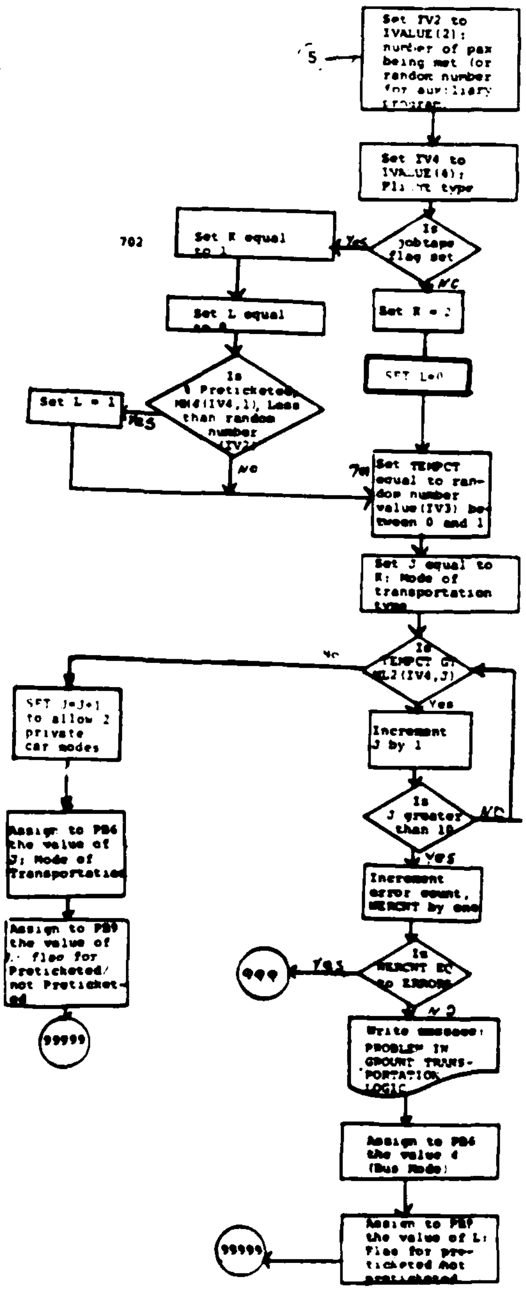
BAGCLAIM


3
Set NPTFM to current point number
Set IV3 to flight table row number
Set J to row number in MH9 for baggage claim area
Set NPTTO to point number of baggage claim area number
Assign statement number 309 to NEXT
950
309
Set PE2 to NPTTO; point number
Set PE4 to 4; process code for baggage claim area
Set PE7 to J; MH9 row for baggage claim area
99999

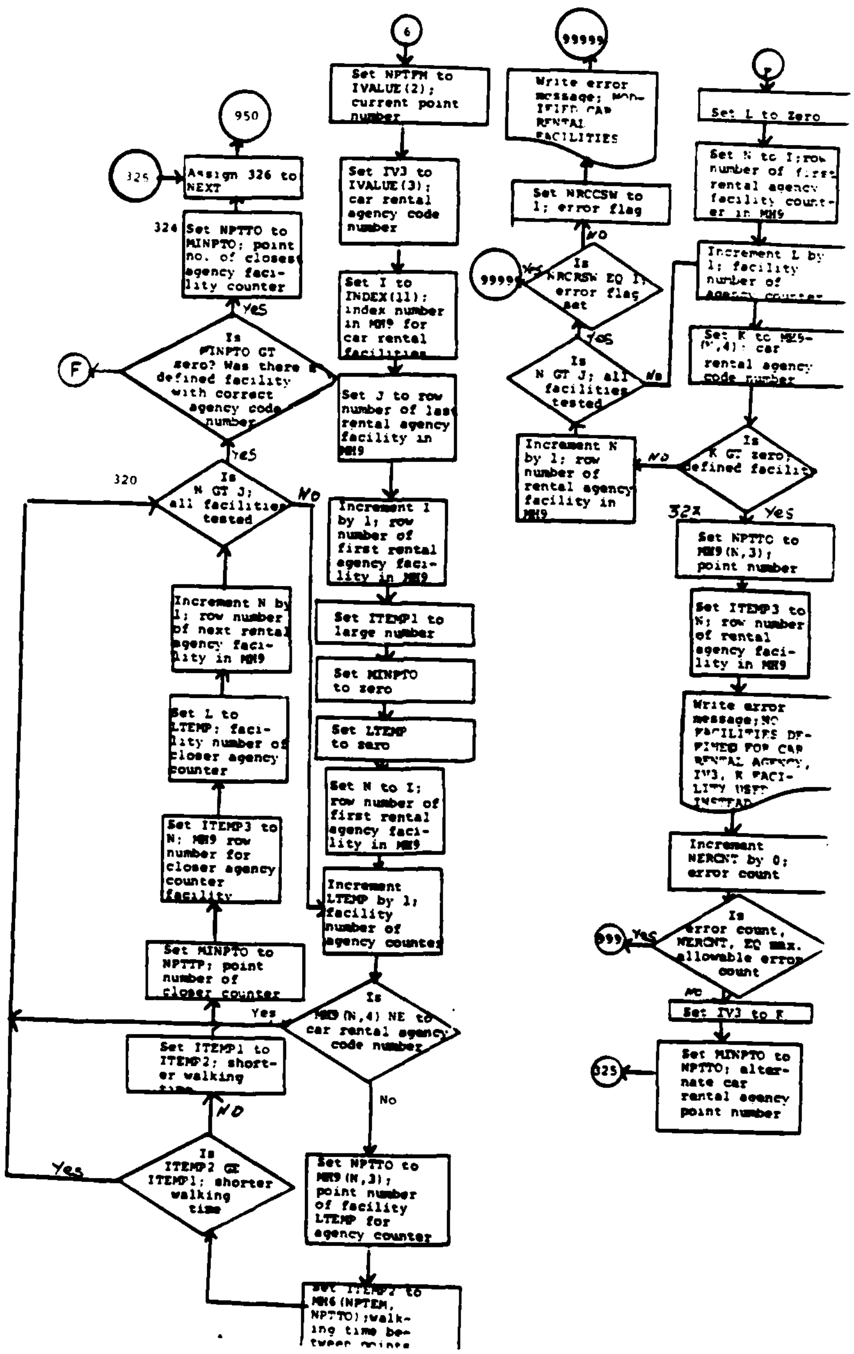
## CUSTOMS


4
Set NPTFM to current point number
Set IV3 to MH9 row number for immigration facility
Set L to associated customs facility number; from MH9(IV3,4)
Set J to MH9 row number for associated customs facility
Set NPTTO to point number of associated customs facility
Assign statement number 313 to NEXT
950
313
Set M to storage, queue number for associated customs facility
Set PH2 to NPTTO; point number
Set PH5 to M; storage,queue number for associated customs
Set PH7 to J; MH9 row number
Set PB5 to 11: Process code for customs
99999

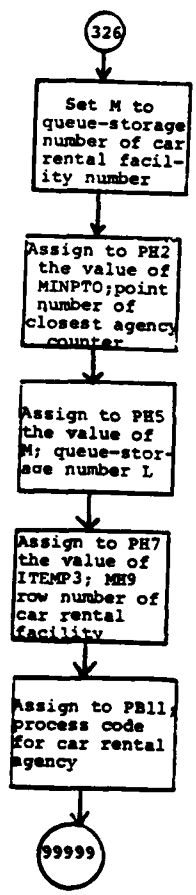
## GROUND TRANSPORT MODE



## RENTACAR

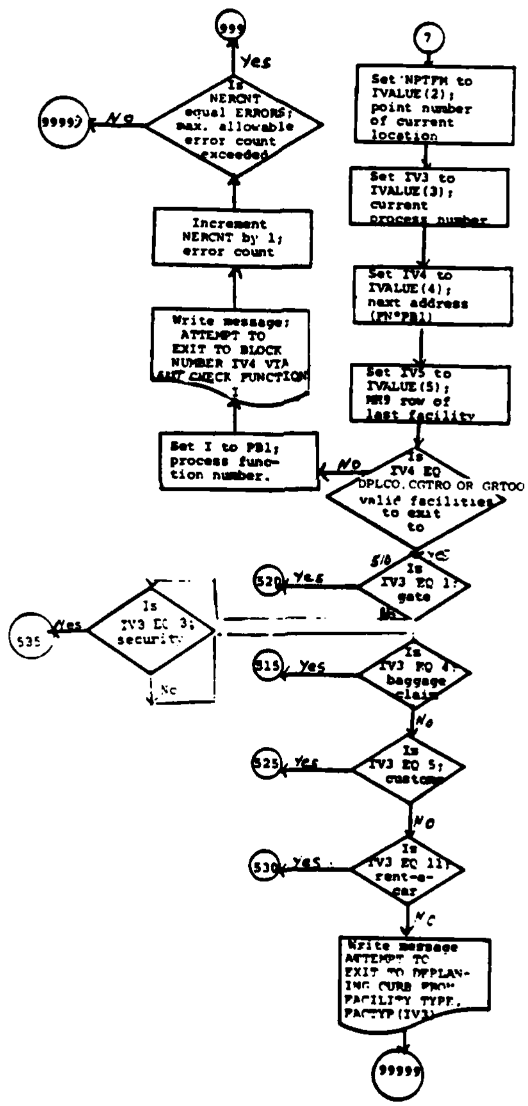


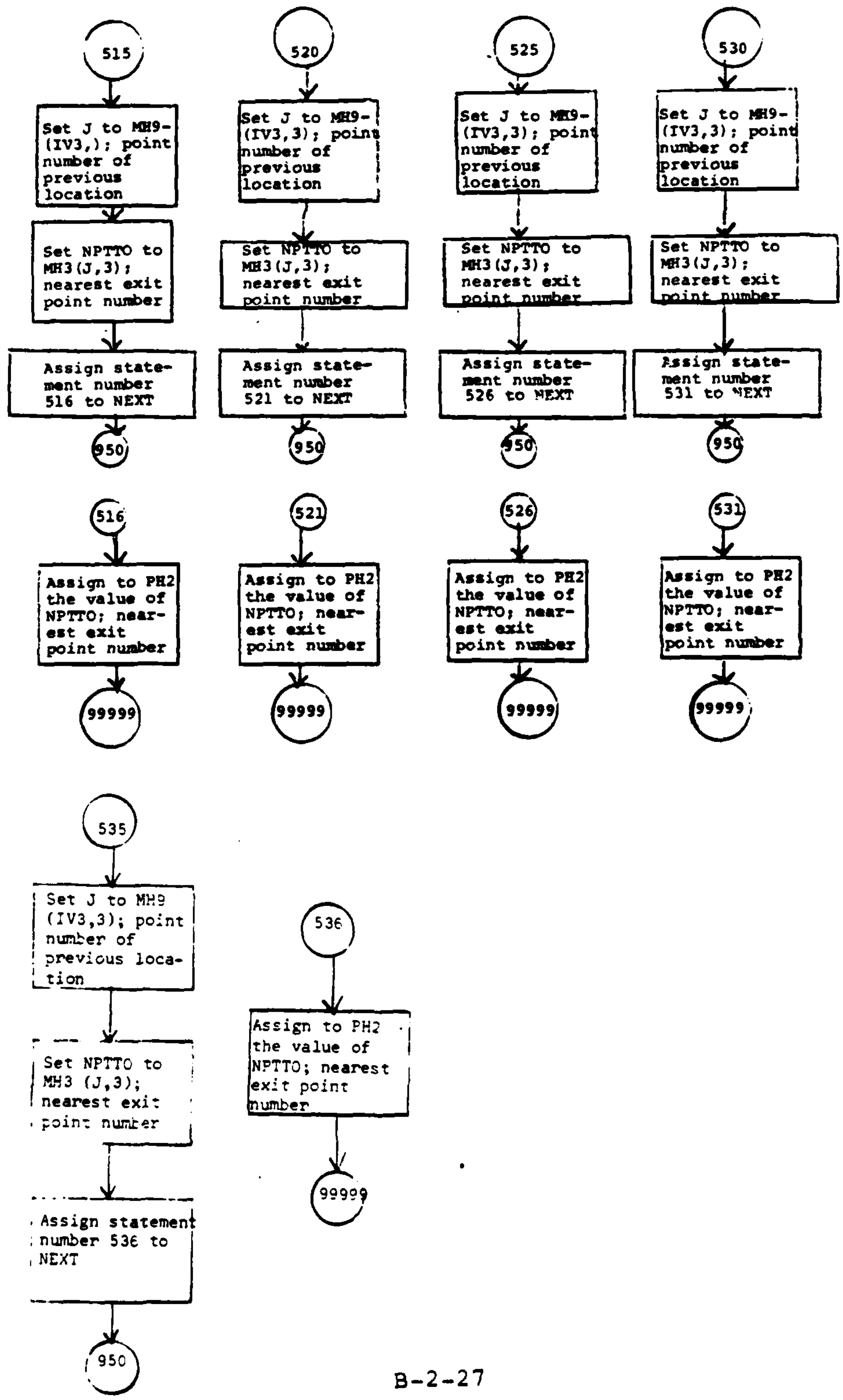
950
325
Assign 326 to NEXT
324
Set NPTTO to MINPTO; point no. of closest agency facility counter
Yes
Is MINPTO GT zero? Was there a defined facility with correct agency code number
F
Yes
320
Is N GT J; all facilities tested
No
Increment N by 1; row number of next rental agency facility in MM9
Set L to LTEMP; facility number of closer agency counter
Set ITEMP3 to N; MM9 row number for closer agency counter facility
Set MINPTO to NPTTP; point number of closer counter
Yes
Set ITEMP1 to ITEMP2; shorter walking time
NO
Yes
Is ITEMP2 GE ITEMP1; shorter walking time
6
Set NPTFM to IVALUE(2); current point number
Set IV3 to IVALUE(3); car rental agency code number
Set I to INDEX(11); index number in MM9 for car rental facilities
Set J to row number of last rental agency facility in MM9
Increment I by 1; row number of first rental agency facility in MM9
Set ITEMP1 to large number
Set MINPTO to zero
Set LTEMP to zero
Set N to I; row number of first rental agency facility in MM9
Increment LTEMP by 1; facility number of agency counter
Is MM9(N,4) NE to car rental agency code number
No
Set NPTTO to MM9(N,3); point number of facility LTEMP for agency counter
Set ITEMP2 to MM6(NPTFM, NPTTO); walking time between points
99999
Write error message; NON-EXISTENT CAR RENTAL FACILITIES
Set NRCCSW to 1; error flag
NO
9999
Yes
Is NRCRSW EQ 1; error flag set
Yes
Is N GT J; all facilities tested
No
Increment N by 1; row number of rental agency facility in MM9
NO
F
Set L to zero
Set N to I; row number of first rental agency facility counter in MM9
Increment L by 1; facility number of agency counter
Set K to MM9(N,4); car rental agency code number
Is K GT zero; defined facility
322
Yes
Set NPTTO to MM9(N,3); point number
Set ITEMP3 to N; row number of rental agency facility in MM9
Write error message; NO FACILITIES DEFINED FOR CAR RENTAL AGENCY, IV3, K FACILITY USED INSTEAD
Increment NERCNT by 0; error count
999
Yes
Is error count, NERCNT, EQ max. allowable error count
NO
Set IV3 to K
325
Set MINPTO to NPTTO; alternate car rental agency point number



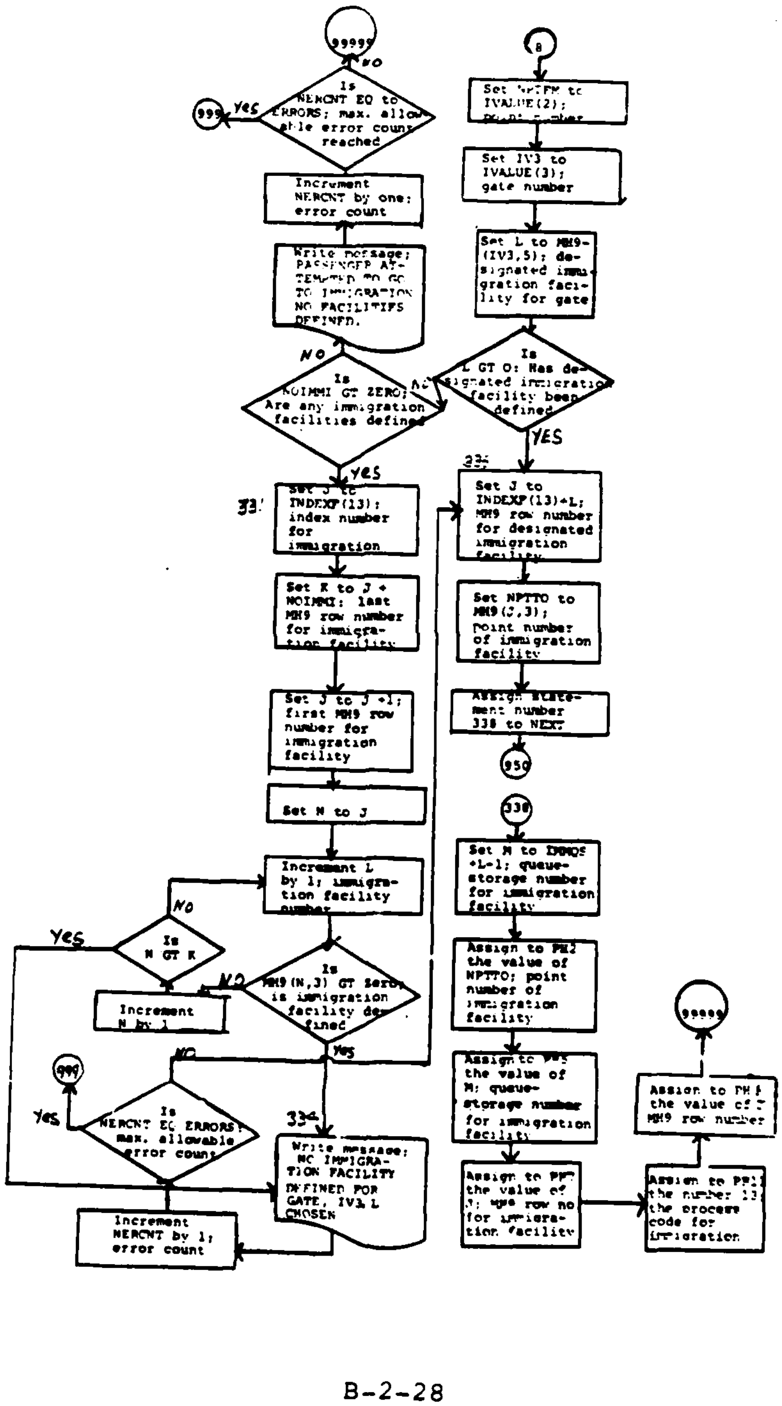
326
Set M to queue-storage number of car rental facility number
Assign to PH2 the value of MINPTO;point number of closest agency counter
Assign to PH5 the value of M; queue-storage number L
Assign to PH7 the value of ITEMP3; MH9 row number of car rental facility
Assign to PB11, process code for car rental agency
99999

EXIT

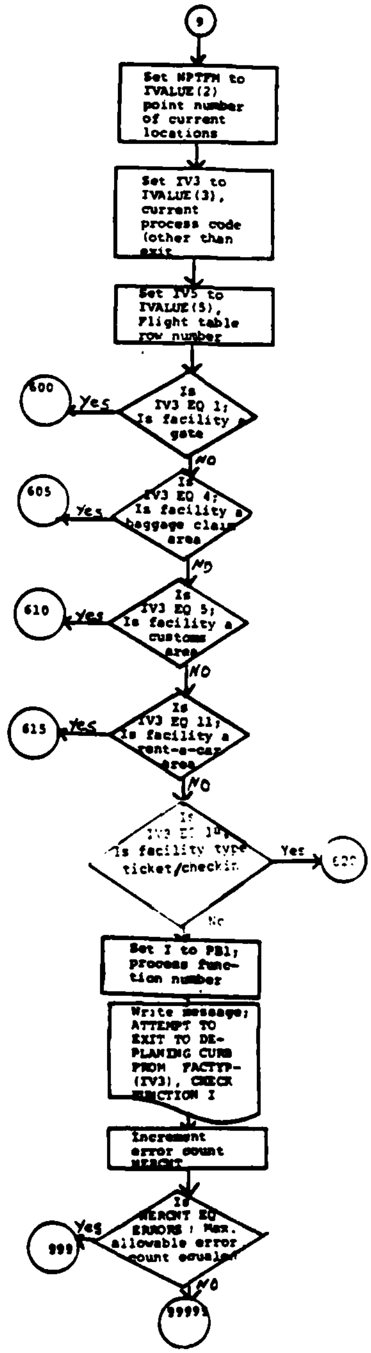

999
Yes
Is NERCNT equal ERRORS; max. allowable error count exceeded
99997
No
Increment NERCNT by 1; error count
Write message; ATTEMPT TO EXIT TO BLOCK NUMBER IV4 VIA EXIT CHECK FUNCTION I
Set I to PB1; process function number.
7
Set NPTFM to IVALUE(2); point number of current location
Set IV3 to IVALUE(3); current process number
Set IV4 to IVALUE(4); next address (FN*PB1)
Set IV5 to IVALUE(5); MM9 row of last facility
Is IV4 EQ DPLCO, CGTRO OR GRTOO valid facilities to exit to
No
Yes
510
Is IV3 EQ 1; gate
520
Yes
No
Is IV3 EQ 3; security
535
Yes
No
Is IV3 EQ 4; baggage claim
515
Yes
No
Is IV3 EQ 5; customs
525
Yes
No
Is IV3 EQ 11; rent-a-car
530
Yes
No
Write message ATTEMPT TO EXIT TO DEPLANING CURB FROM FACILITY TYPE, FACTYP(IV3)
99999

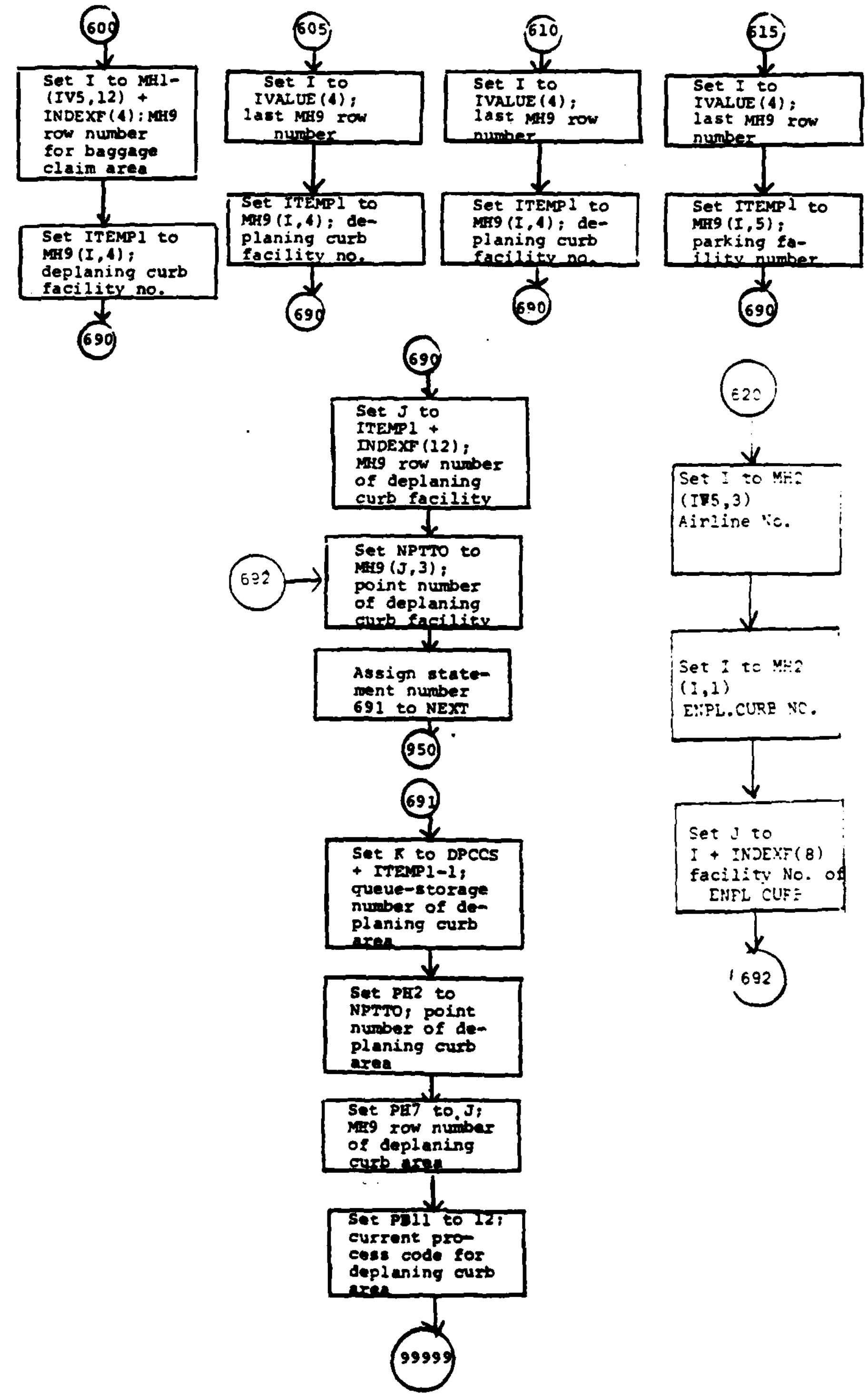

515
Set J to MH9-(IV3,); point number of previous location
Set NPTTO to MH3(J,3); nearest exit point number
Assign statement number 516 to NEXT
950
520
Set J to MH9-(IV3,3); point number of previous location
Set NPTTO to MH3(J,3); nearest exit point number
Assign statement number 521 to NEXT
950
525
Set J to MH9-(IV3,3); point number of previous location
Set NPTTO to MH3(J,3); nearest exit point number
Assign statement number 526 to NEXT
950
530
Set J to MH9-(IV3,3); point number of previous location
Set NPTTO to MH3(J,3); nearest exit point number
Assign statement number 531 to NEXT
950
516
Assign to PH2 the value of NPTTO; nearest exit point number
99999
521
Assign to PH2 the value of NPTTO; nearest exit point number
99999
526
Assign to PH2 the value of NPTTO; nearest exit point number
99999
531
Assign to PH2 the value of NPTTO; nearest exit point number
99999
535
Set J to MH9 (IV3,3); point number of previous location
Set NPTTO to MH3 (J,3); nearest exit point number
Assign statement number 536 to NEXT
950
536
Assign to PH2 the value of NPTTO; nearest exit point number
99999

## IMMIGRATION



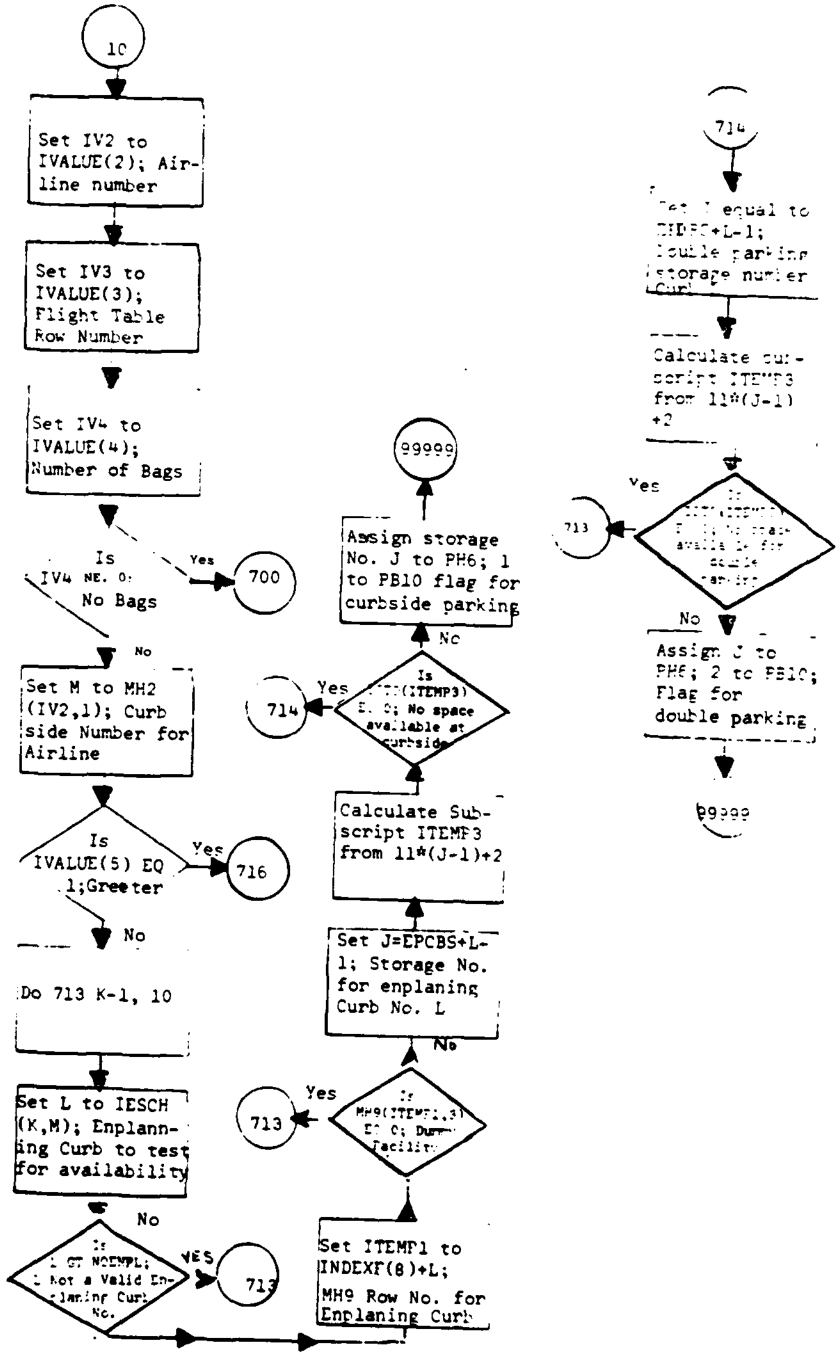
DEPLANING CURB (PAX)

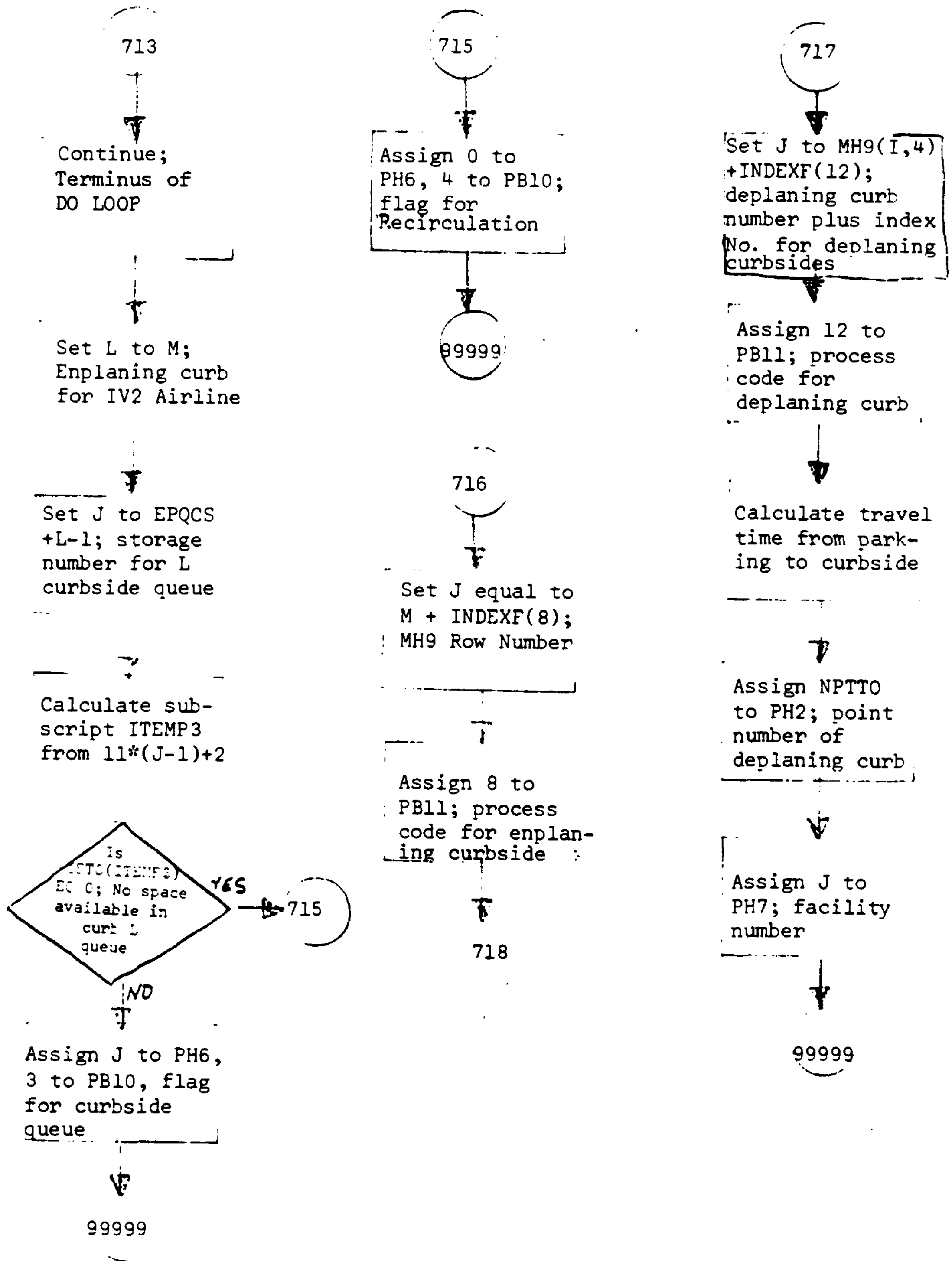




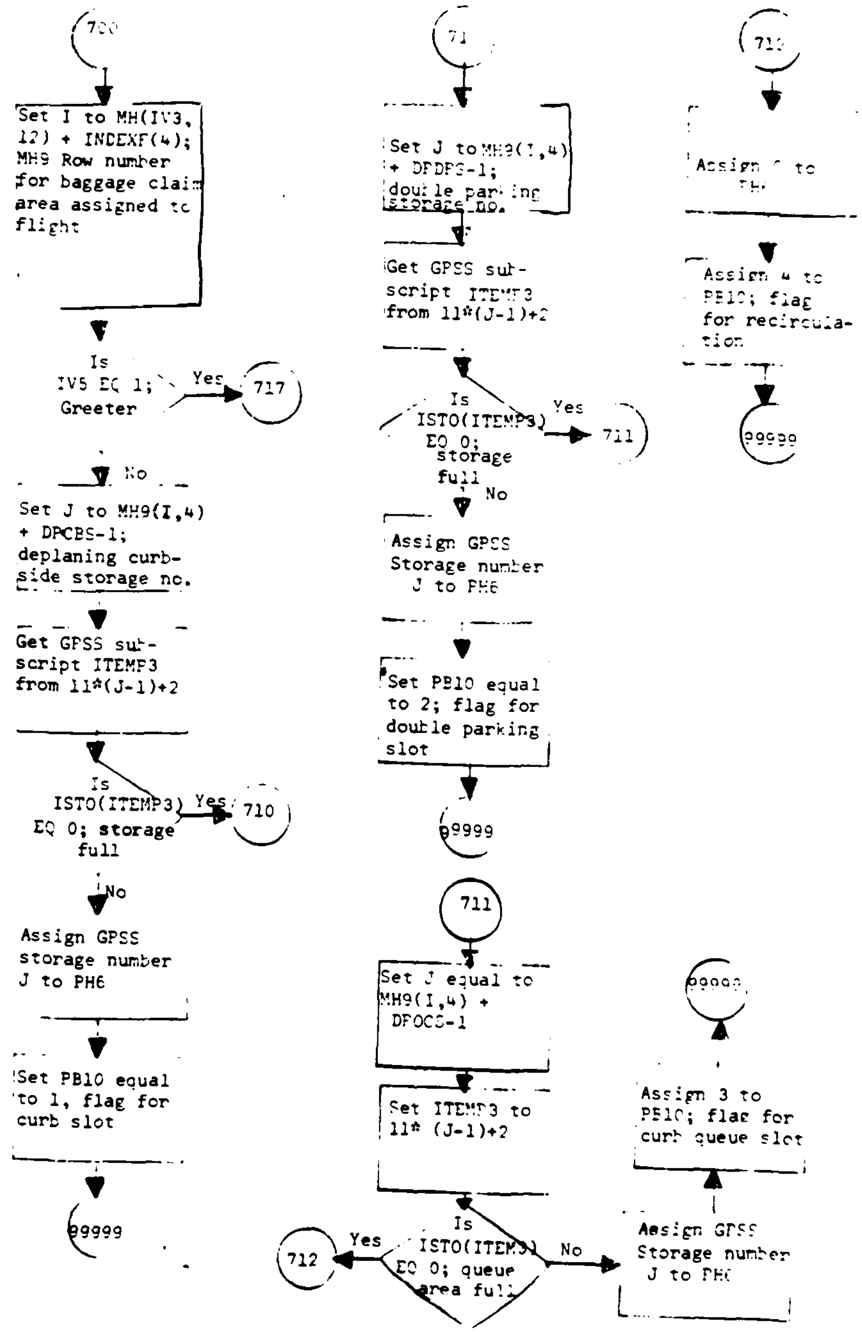
600
Set I to MH1-(IV5,12) + INDEXF(4):MH9 row number for baggage claim area
Set ITEMP1 to MH9(I,4); deplaning curb facility no.
690
605
Set I to IVALUE(4); last MH9 row number
Set ITEMP1 to MH9(I,4); deplaning curb facility no.
690
610
Set I to IVALUE(4); last MH9 row number
Set ITEMP1 to MH9(I,4); deplaning curb facility no.
690
615
Set I to IVALUE(4); last MH9 row number
Set ITEMP1 to MH9(I,5); parking facility number
690
690
Set J to ITEMP1 + INDEXF(12); MH9 row number of deplaning curb facility
692
Set NPTTO to MH9(J,3); point number of deplaning curb facility
Assign statement number 691 to NEXT
950
691
Set K to DPCCS + ITEMP1-1; queue-storage number of deplaning curb area
Set PH2 to NPTTO; point number of deplaning curb area
Set PH7 to J; MH9 row number of deplaning curb area
Set PB11 to 12; current process code for deplaning curb area
99999
620
Set I to MH2 (IV5,3) Airline No.
Set I to MH2 (I,1) ENPL.CURB NO.
Set J to I + INDEXF(8) facility No. of ENPL CURB
692

## DEPLANING CURB (CARS AND GREETERS



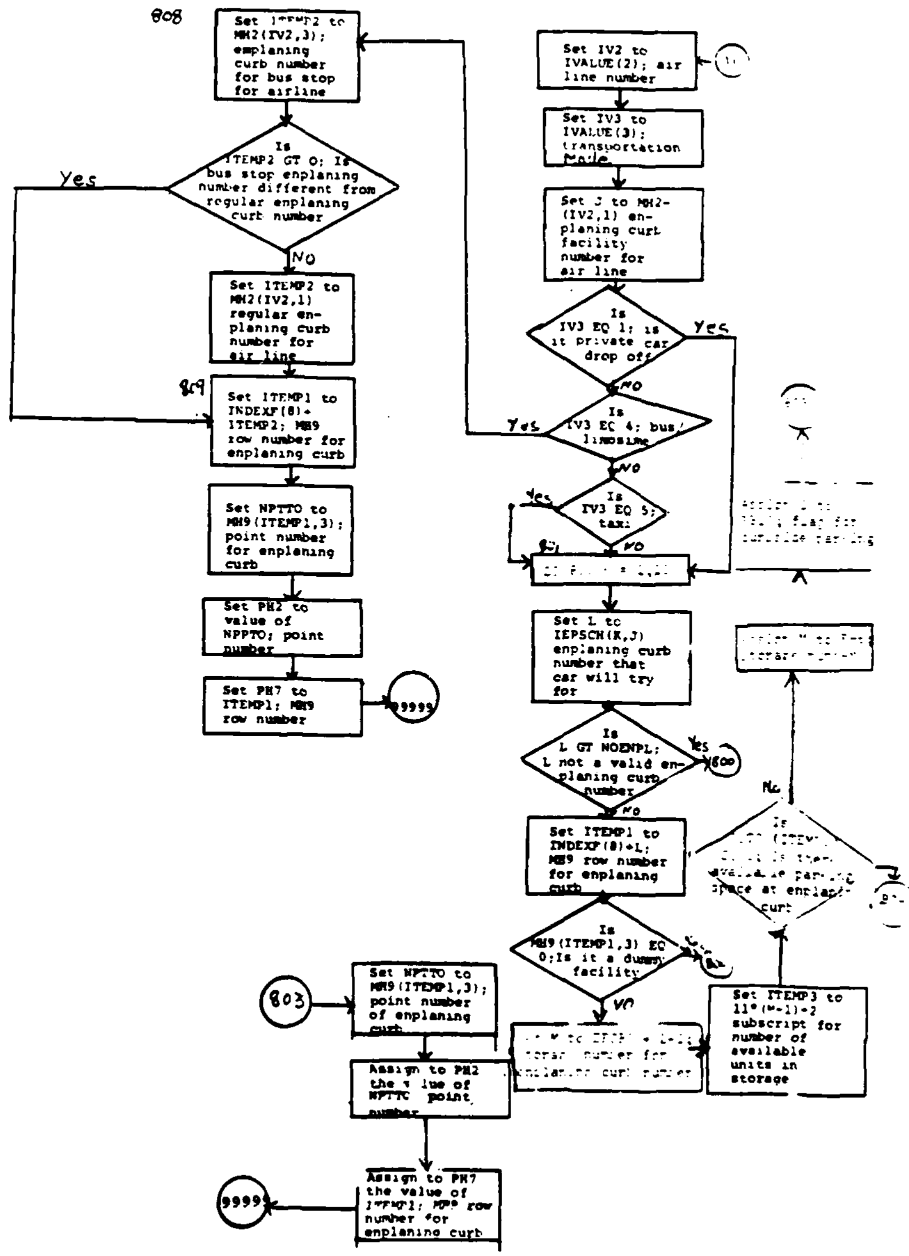


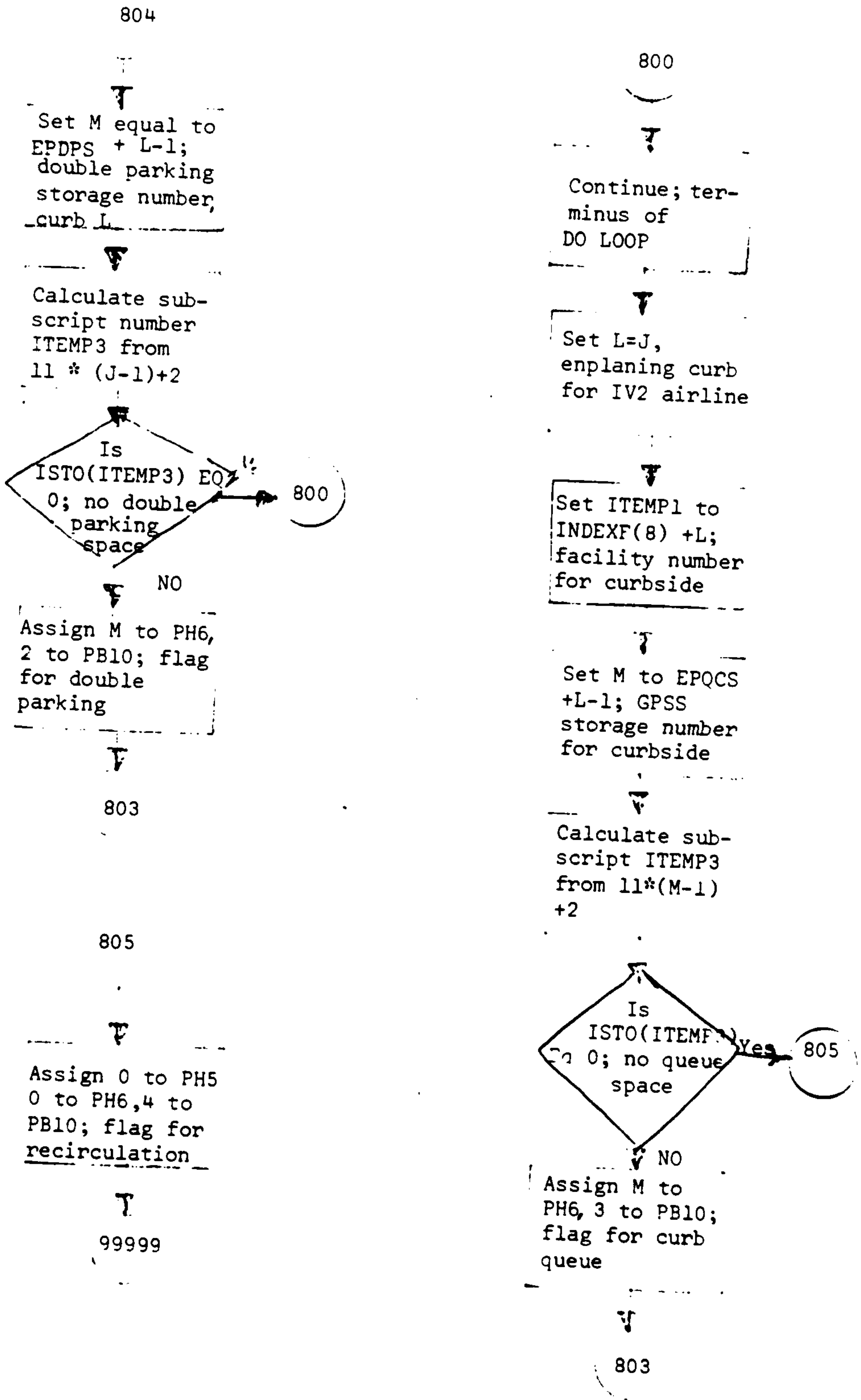
713
Continue; Terminus of DO LOOP
Set L to M; Enplaning curb for IV2 Airline
Set J to EPQCS +L-1; storage number for L curbside queue
Calculate sub-script ITEMP3 from 11*(J-1)+2
Is
No space available in
queue
YES
715
NO
Assign J to PH6, 3 to PB10, flag for curbside queue
99999
715
Assign 0 to PH6, 4 to PB10; flag for Recirculation
99999
716
Set J equal to M + INDEXF(8); MH9 Row Number
Assign 8 to PB11; process code for enplan-ing curbside
718
717
Set J to MH9(I,4) +INDEXF(12); deplaning curb number plus index No. for deplaning curbsides
Assign 12 to PB11; process code for deplaning curb
Calculate travel time from park-ing to curbside
Assign NPTTO to PH2; point number of deplaning curb
Assign J to PH7; facility number
99999



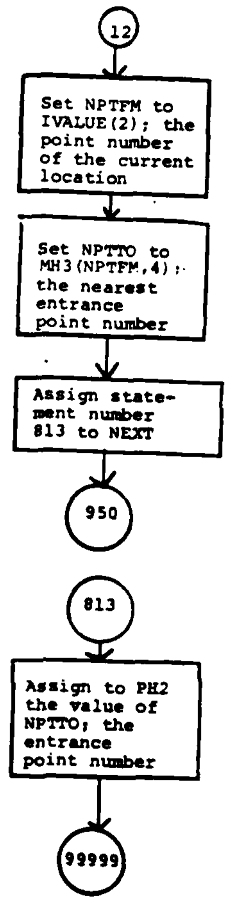
720
Set I to MH(IV3, 12) + INDEXF(4); MH9 Row number for baggage claim area assigned to flight
Is IV5 EQ 1; Greeter
Yes
717
No
Set J to MH9(I,4) + DPCBS-1; deplaning curb-side storage no.
Get GPSS sub-script ITEMP3 from 11*(J-1)+2
Is ISTO(ITEMP3) EQ 0; storage full
Yes
710
No
Assign GPSS storage number J to PH6
Set PB10 equal to 1, flag for curb slot
99999
710
Set J to MH9(I,4) + DPDPS-1; double parking storage no.
Get GPSS sub-script ITEMP3 from 11*(J-1)+2
Is ISTO(ITEMP3) EQ 0; storage full
Yes
711
No
Assign GPSS Storage number J to PH6
Set PB10 equal to 2; flag for double parking slot
99999
711
Set J equal to MH9(I,4) + DPOCS-1
Set ITEMP3 to 11* (J-1)+2
Is ISTO(ITEMP3) EQ 0; queue area full
Yes
712
No
Assign GPSS Storage number J to PH6
Assign 3 to PB10; flag for curb queue slot
99999
712
Assign 4 to PB10; flag for recirculation
99999

## ENPLANING CURB

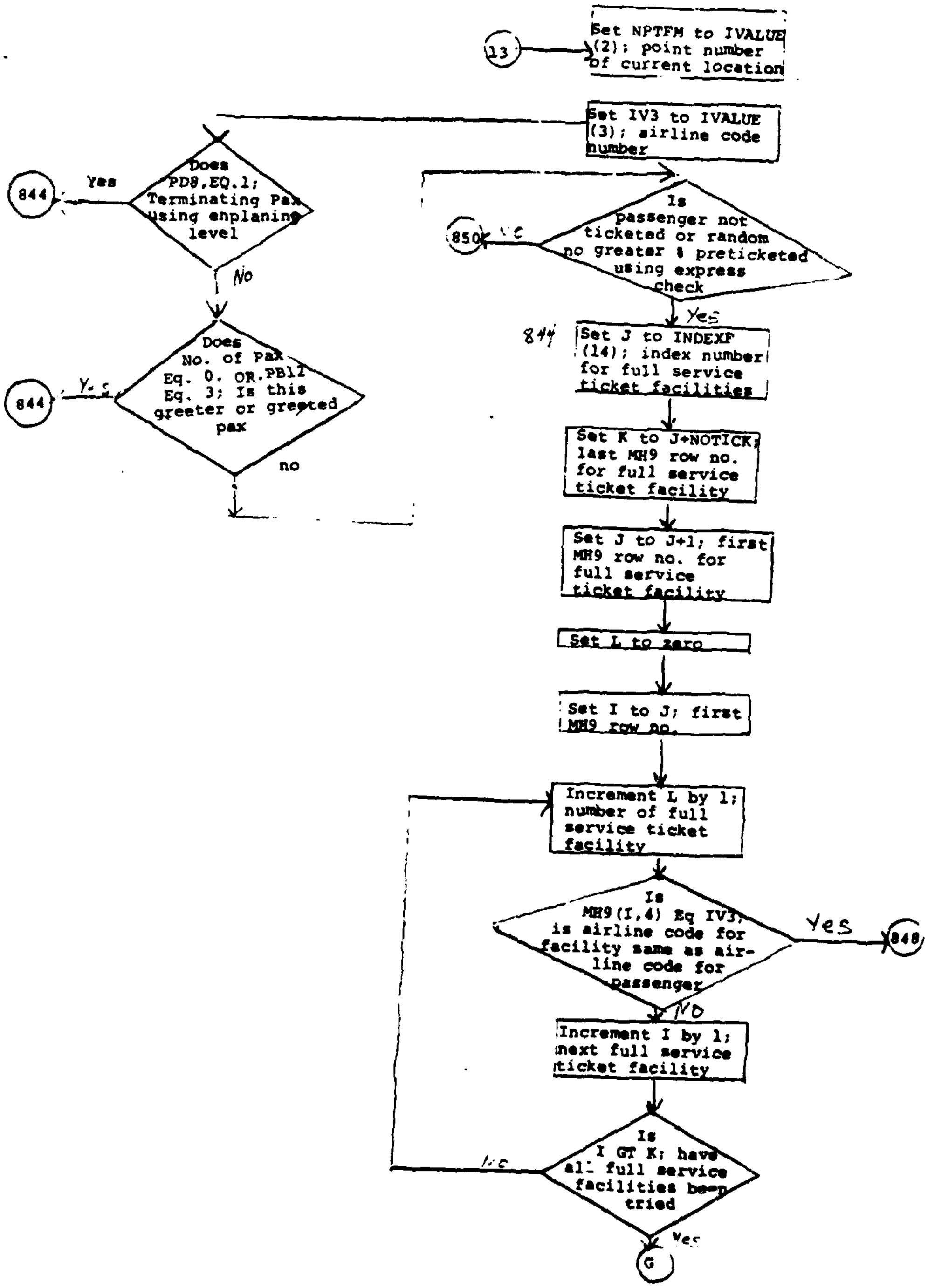


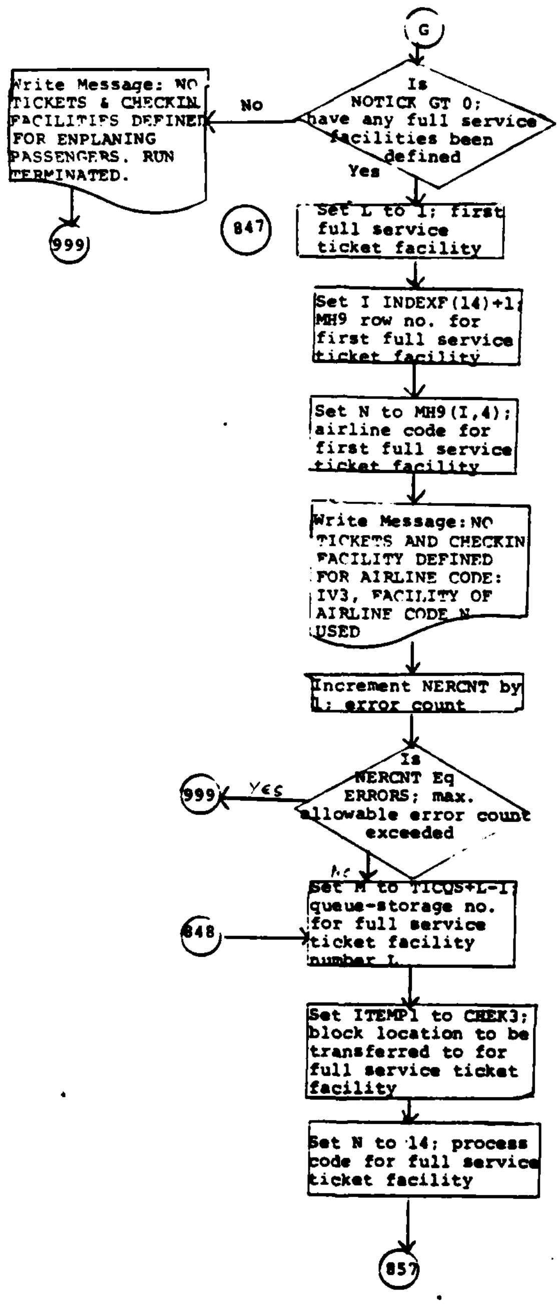

804
Set M equal to EPDPS + L-1; double parking storage number, curb L
Calculate subscript number ITEMP3 from 11 * (J-1)+2
Is ISTO(ITEMP3) EQ 0; no double parking space
800
NO
Assign M to PH6, 2 to PB10; flag for double parking
803
805
Assign 0 to PH5 0 to PH6, 4 to PB10; flag for recirculation
99999
800
Continue; terminus of DO LOOP
Set L=J, enplaning curb for IV2 airline
Set ITEMP1 to INDEXF(8) +L; facility number for curbside
Set M to EPQCS +L-1; GPSS storage number for curbside
Calculate subscript ITEMP3 from 11*(M-1) +2
Is ISTO(ITEMP3) Eq 0; no queue space
Yes
805
NO
Assign M to PH6, 3 to PB10; flag for curb queue
803

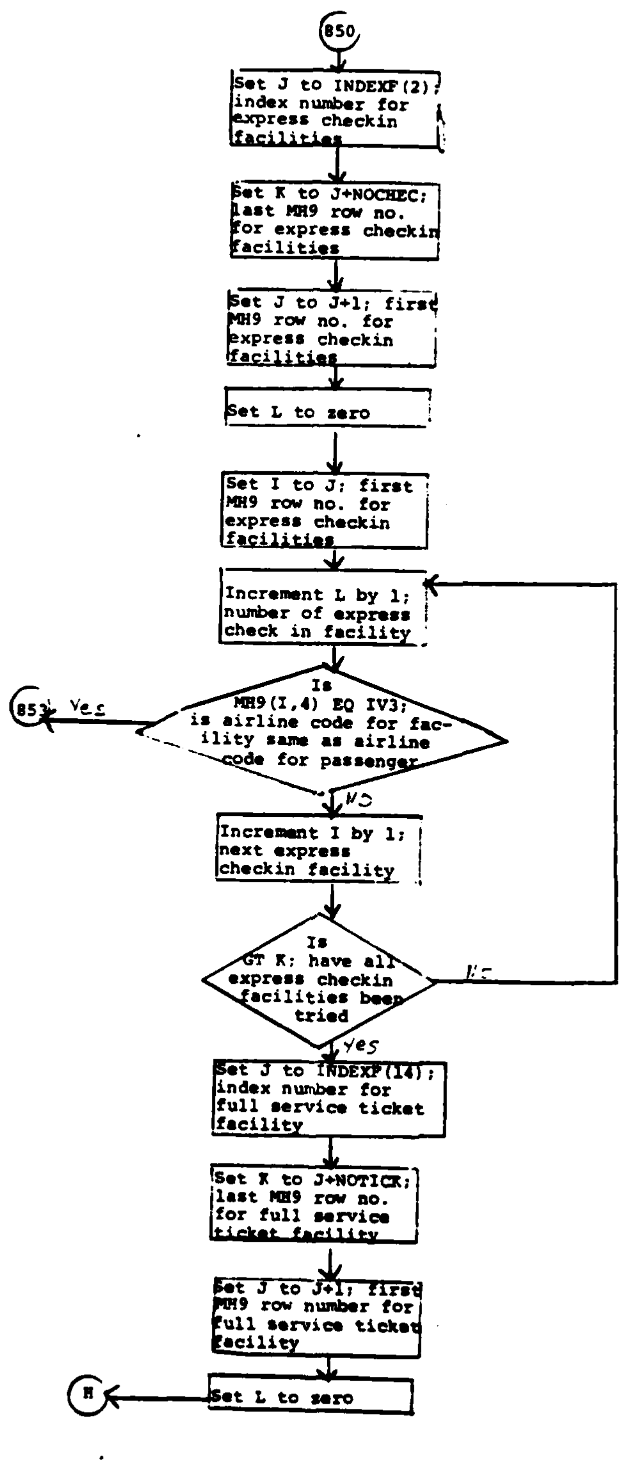
# ENTRANCE

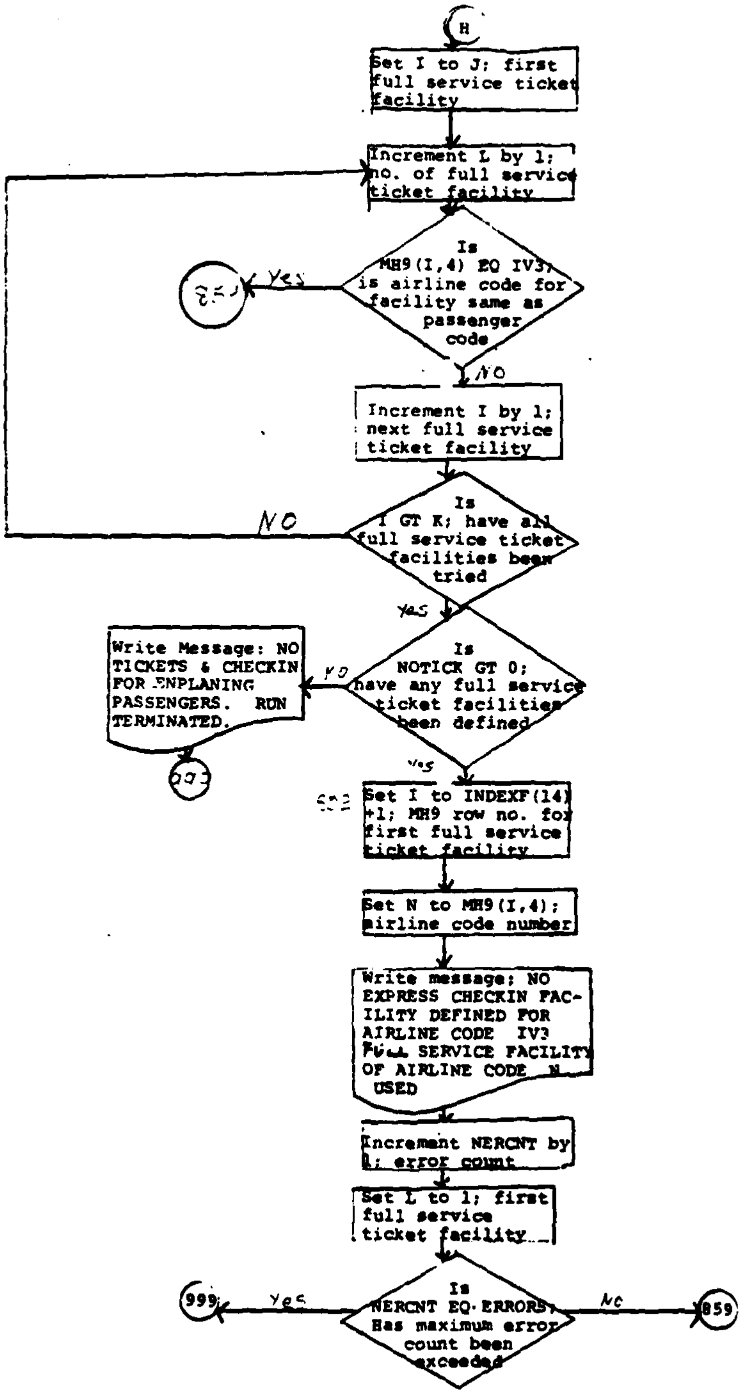

12
Set NPTFM to IVALUE(2); the point number of the current location
Set NPTTO to MH3(NPTFM,4); the nearest entrance point number
Assign statement number 813 to NEXT
950
813
Assign to PH2 the value of NPTTO; the entrance point number
99999

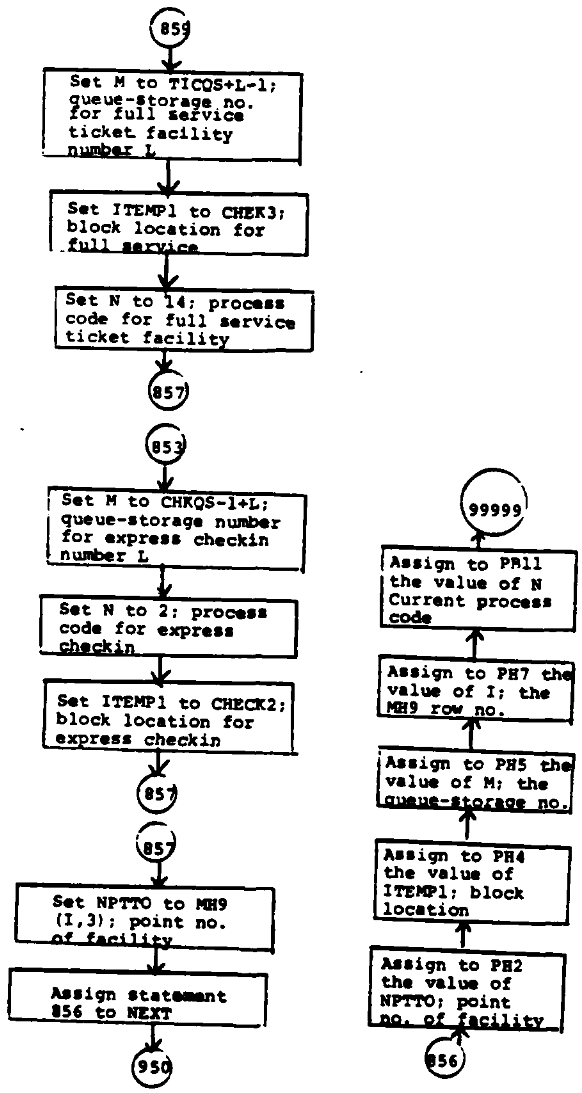
## TICKETING AND CHECKIN



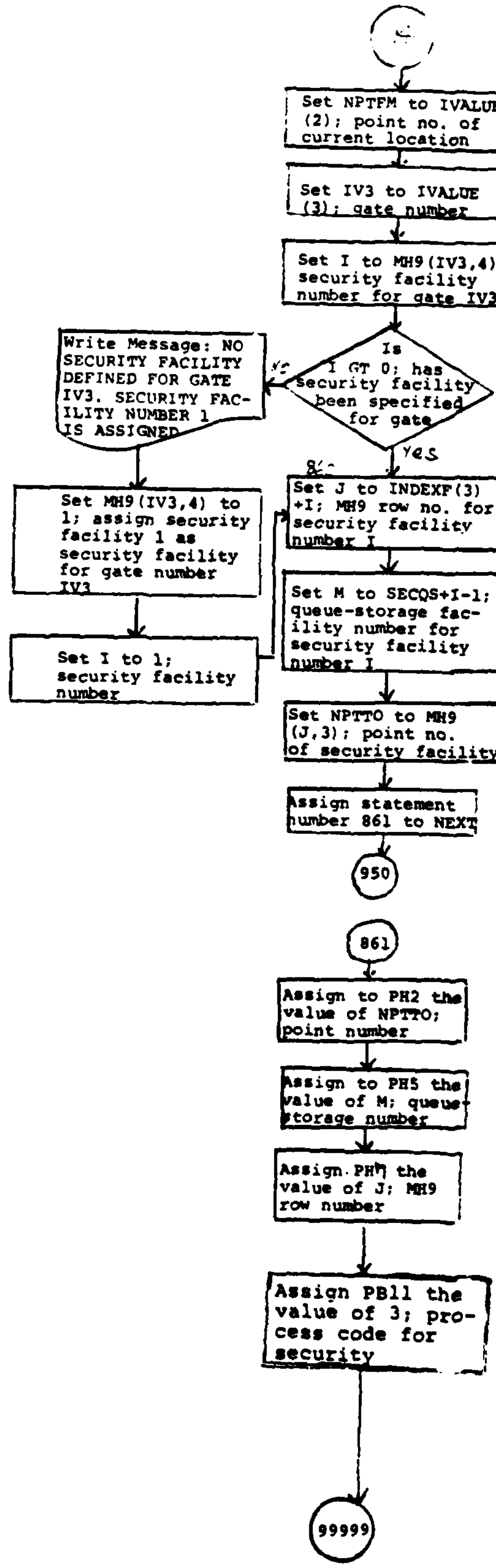

G
Is
NOTICK GT 0;
have any full service
facilities been
defined
No
Yes
Write Message: NO
TICKETS & CHECKIN
FACILITIES DEFINED
FOR ENPLANING
PASSENGERS. RUN
TERMINATED.
999
847
Set L to 1; first
full service
ticket facility
Set I INDEXF(14)+1;
MH9 row no. for
first full service
ticket facility
Set N to MH9(I,4);
airline code for
first full service
ticket facility
Write Message: NO
TICKETS AND CHECKIN
FACILITY DEFINED
FOR AIRLINE CODE:
IV3, FACILITY OF
AIRLINE CODE N
USED
Increment NERCNT by
1; error count
Is
NERCNT Eq
ERRORS; max.
allowable error count
exceeded
Yes
999
No
Set M to TICQS+L-1;
queue-storage no.
for full service
ticket facility
number L.
848
Set ITEMP1 to CHEK3;
block location to be
transferred to for
full service ticket
facility
Set N to 14; process
code for full service
ticket facility
857


850
Set J to INDEXF(2); index number for express checkin facilities
Set K to J+NOCHEC; last MH9 row no. for express checkin facilities
Set J to J+1; first MH9 row no. for express checkin facilities
Set L to zero
Set I to J; first MH9 row no. for express checkin facilities
Increment L by 1; number of express check in facility
Is MH9(I,4) EQ IV3; is airline code for facility same as airline code for passenger
853
Yes
No
Increment I by 1; next express checkin facility
Is I GT K; have all express checkin facilities been tried
No
Yes
Set J to INDEXF(14); index number for full service ticket facility
Set K to J+NOTICK; last MH9 row no. for full service ticket facility
Set J to J+1; first MH9 row number for full service ticket facility
Set L to zero
H


H
Set I to J; first full service ticket facility
Increment L by 1; no. of full service ticket facility
Is MH9(I,4) EQ IV3; is airline code for facility same as passenger code
Yes
NO
Increment I by 1; next full service ticket facility
Is I GT K; have all full service ticket facilities been tried
NO
Yes
Is NOTICK GT 0; have any full service ticket facilities been defined
NO
Write Message: NO TICKETS & CHECKIN FOR ENPLANING PASSENGERS. RUN TERMINATED.
Yes
Set I to INDEXF(14) +1; MH9 row no. for first full service ticket facility
Set N to MH9(I,4); airline code number
Write message; NO EXPRESS CHECKIN FACILITY DEFINED FOR AIRLINE CODE IV3 FULL SERVICE FACILITY OF AIRLINE CODE N USED
Increment NERCNT by 1; error count
Set L to 1; first full service ticket facility
Is NERCNT EQ ERRORS; Has maximum error count been exceeded
Yes
999
NO
859

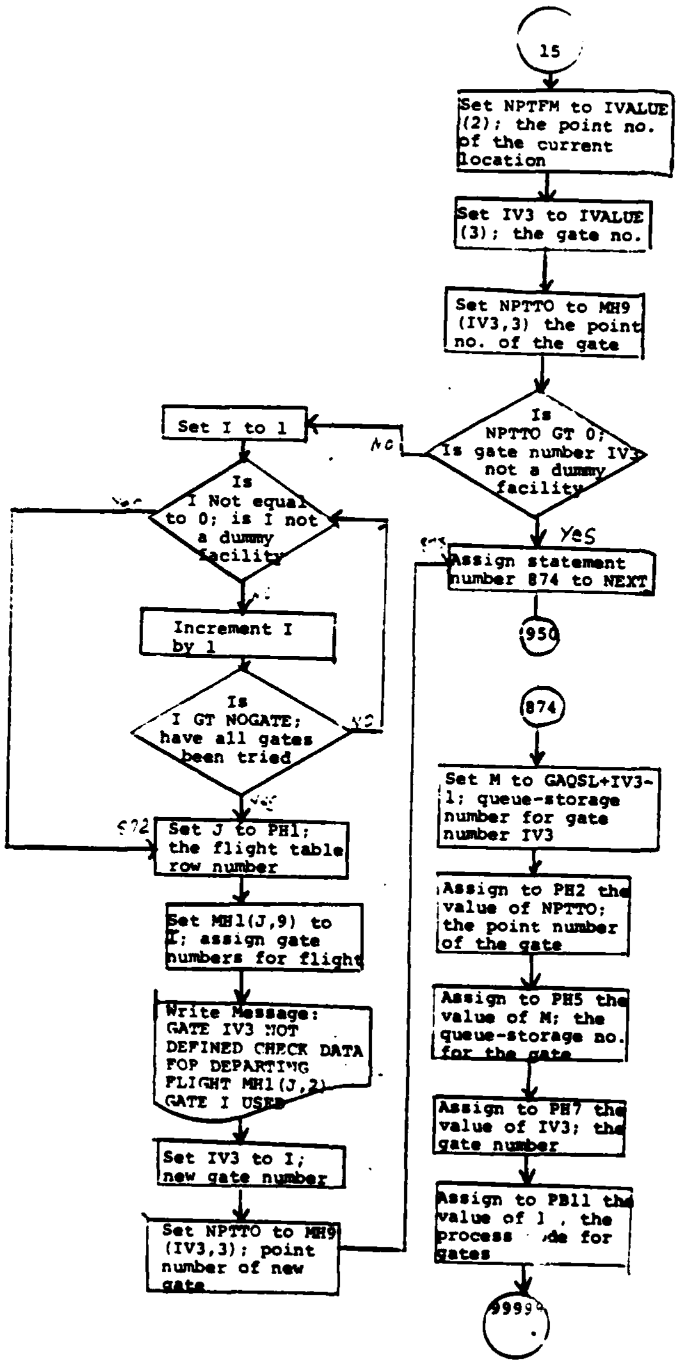

859
Set M to TICQS+L-1; queue-storage no. for full service ticket facility number L
Set ITEMP1 to CHEK3; block location for full service
Set N to 14; process code for full service ticket facility
857
853
Set M to CHKQS-1+L; queue-storage number for express checkin number L
Set N to 2; process code for express checkin
Set ITEMP1 to CHECK2; block location for express checkin
857
857
Set NPTTO to MH9 (I,3); point no. of facility
Assign statement 856 to NEXT
950
99999
Assign to PB11 the value of N Current process code
Assign to PH7 the value of I; the MH9 row no.
Assign to PH5 the value of M; the queue-storage no.
Assign to PH4 the value of ITEMP1; block location
Assign to PH2 the value of NPTTO; point no. of facility
856

## SECURITY

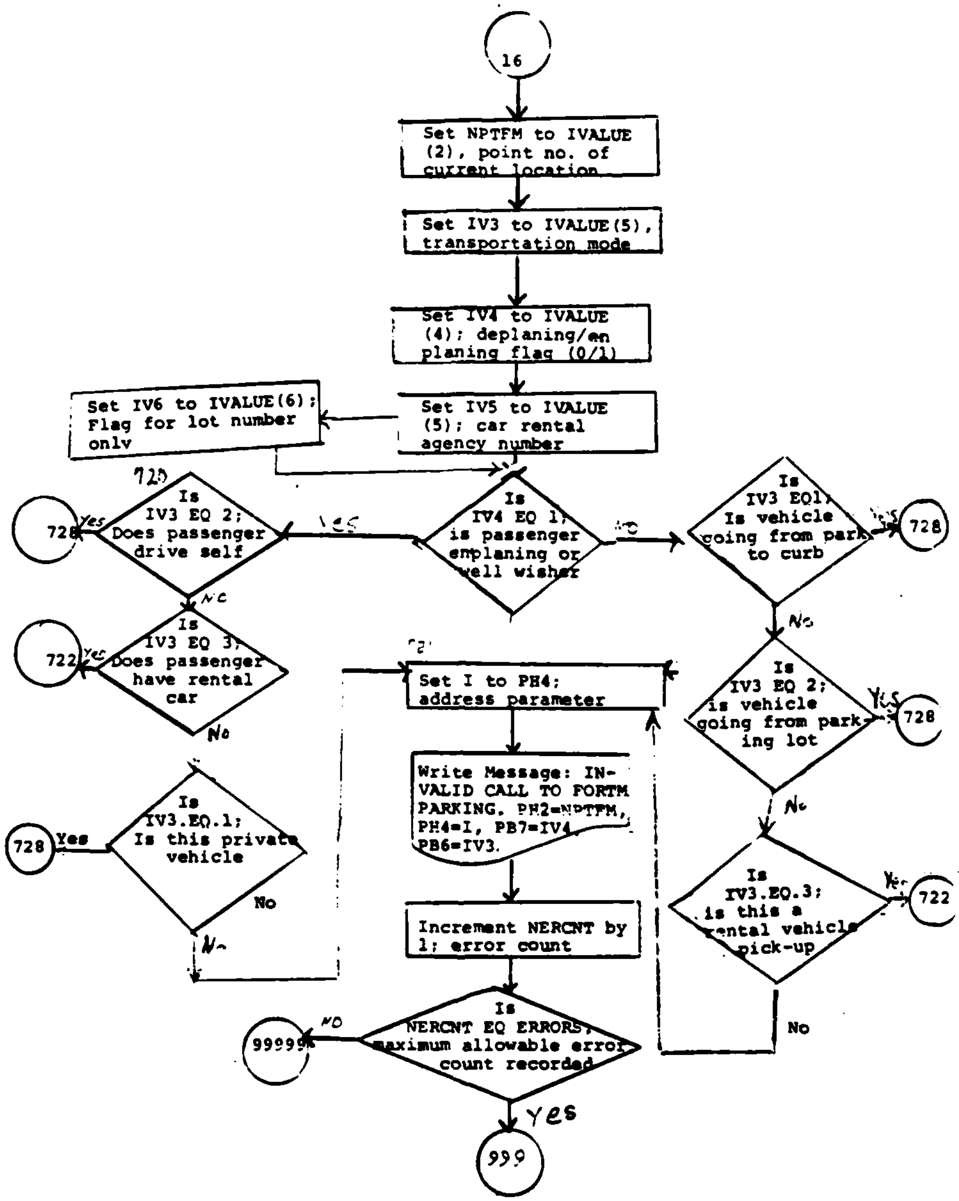


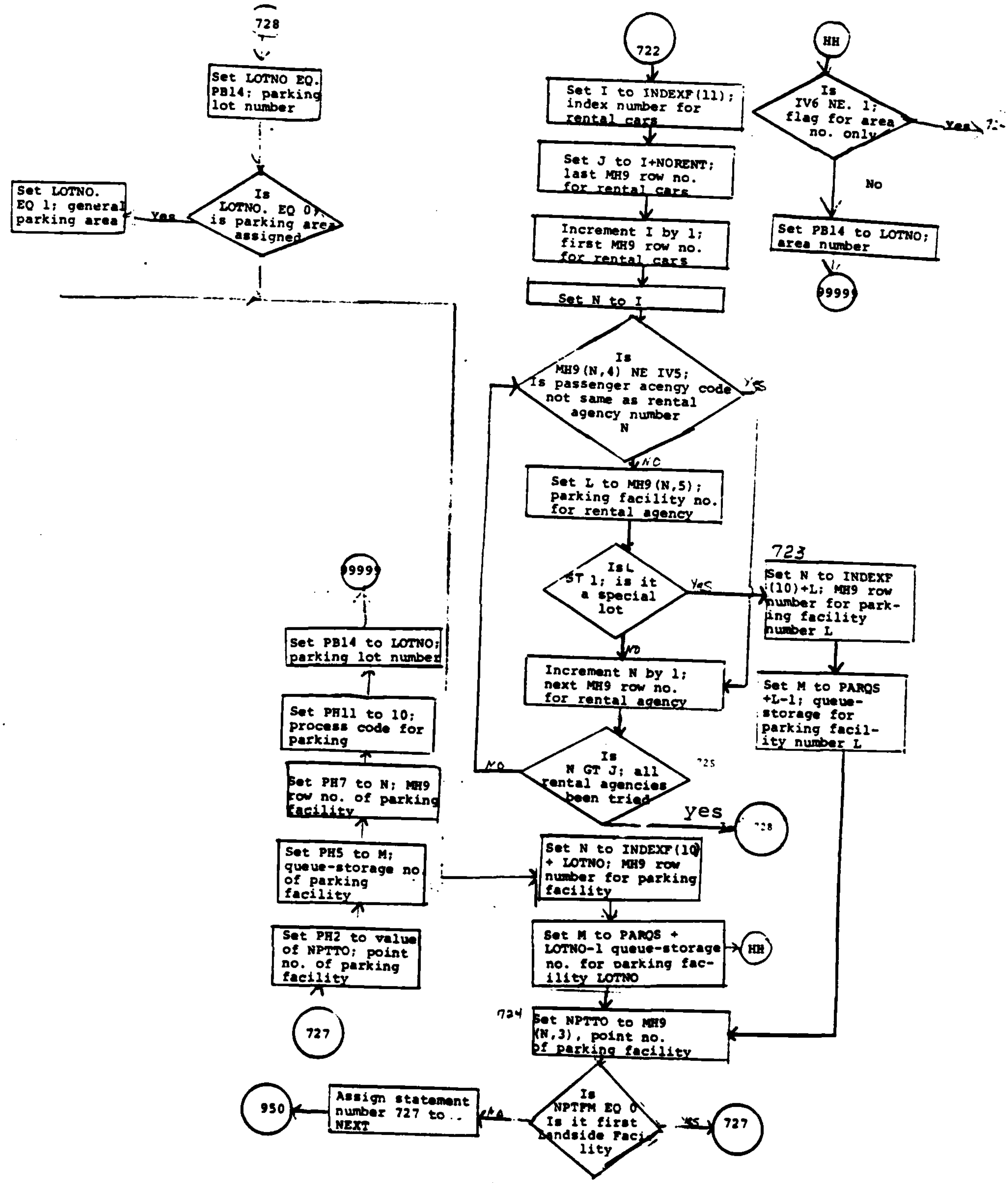
Set NPTFM to IVALUE (2); point no. of current location
Set IV3 to IVALUE (3); gate number
Set I to MH9(IV3,4) security facility number for gate IV3
Is I GT 0; has security facility been specified for gate
No
Yes
Write Message: NO SECURITY FACILITY DEFINED FOR GATE IV3. SECURITY FACILITY NUMBER 1 IS ASSIGNED
Set MH9(IV3,4) to 1; assign security facility 1 as security facility for gate number IV3
Set I to 1; security facility number
Set J to INDEXF(3) +I; MH9 row no. for security facility number I
Set M to SECQS+I-1; queue-storage facility number for security facility number I
Set NPTTO to MH9 (J,3); point no. of security facility
Assign statement number 861 to NEXT
950
861
Assign to PH2 the value of NPTTO; point number
Assign to PH5 the value of M; queue-storage number
Assign PH7 the value of J; MH9 row number
Assign PB11 the value of 3; process code for security
99999

GATE (ENPLANING PAX)

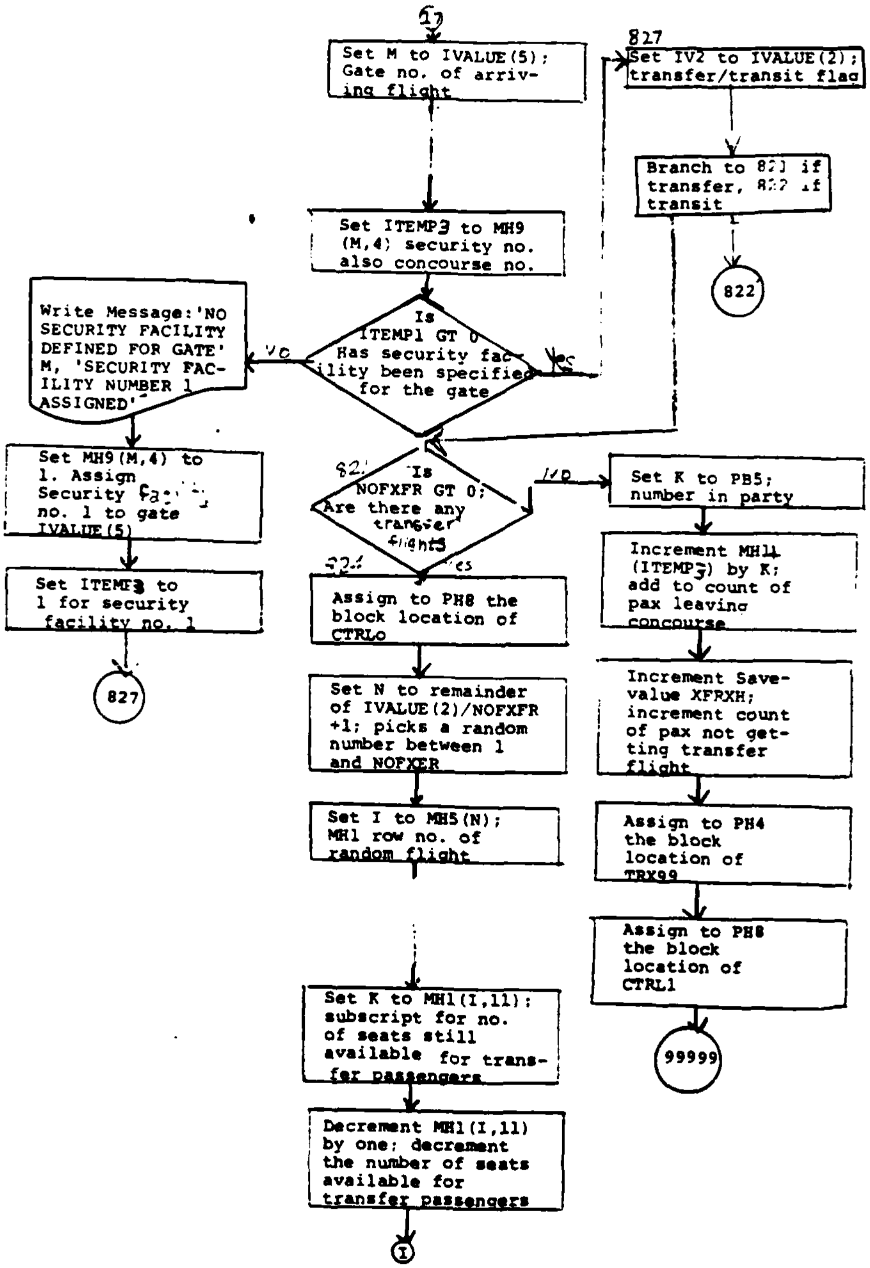


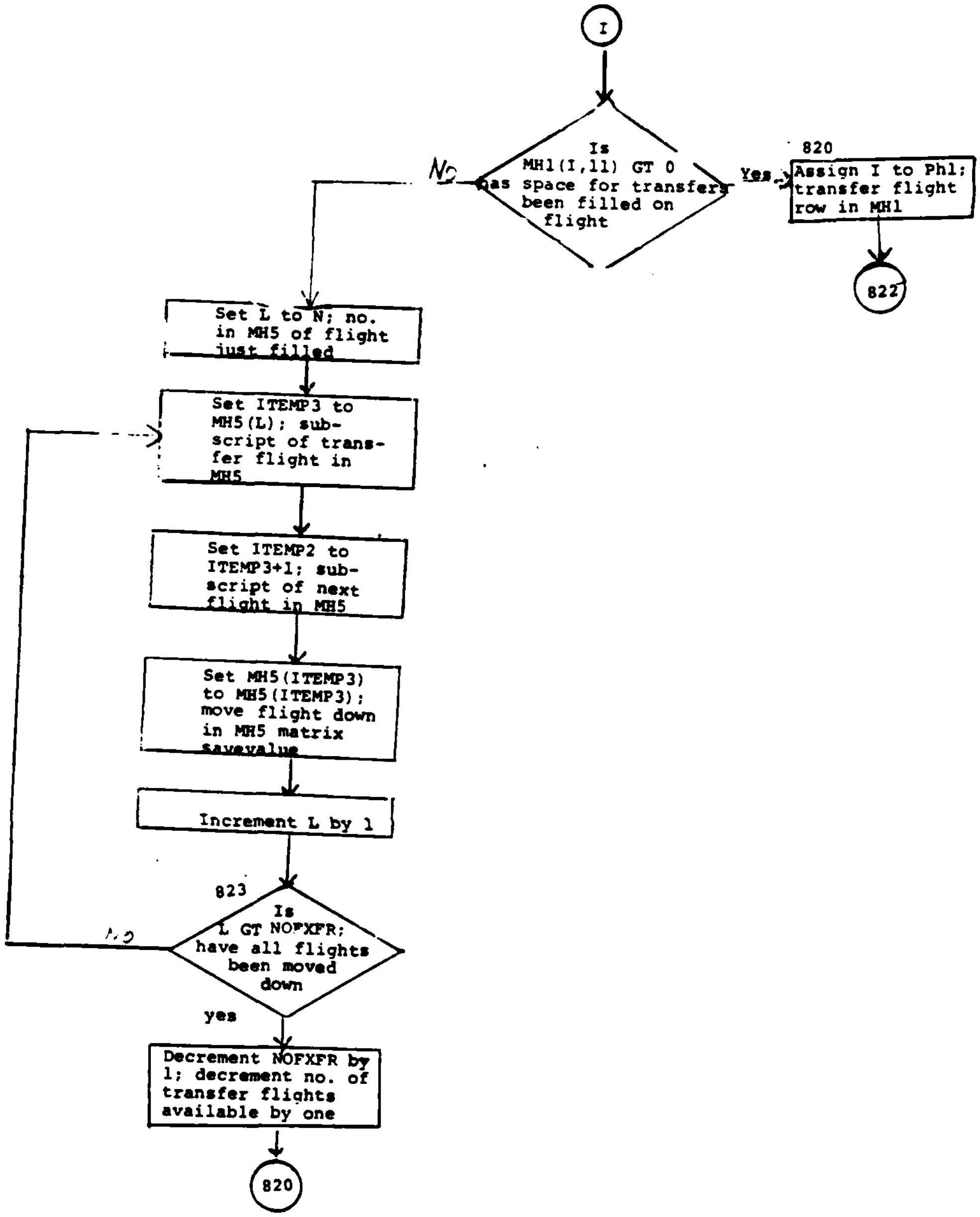
## PARKING (PAX)


16
Set NPTFM to IVALUE (2), point no. of current location
Set IV3 to IVALUE(5), transportation mode
Set IV4 to IVALUE (4); deplaning/en planing flag (0/1)
Set IV5 to IVALUE (5); car rental agency number
Set IV6 to IVALUE(6); Flag for lot number only
720
Is IV3 EQ 2; Does passenger drive self
Yes
728
No
Is IV4 EQ 1; is passenger enplaning or well wisher
Yes
No
Is IV3 EQ1; Is vehicle going from park to curb
Yes
728
No
Is IV3 EQ 3; Does passenger have rental car
Yes
722
No
Is IV3 EQ 2; is vehicle going from parking lot
Yes
728
No
Set I to PH4; address parameter
Write Message: INVALID CALL TO FORTM PARKING. PH2=NPTFM, PH4=I, PB7=IV4, PB6=IV3.
Is IV3.EQ.1; Is this private vehicle
Yes
728
No
Increment NERCNT by 1; error count
Is IV3.EQ.3; is this a rental vehicle pick-up
Yes
722
No
Is NERCNT EQ ERRORS; maximum allowable error count recorded
No
99999
Yes
999

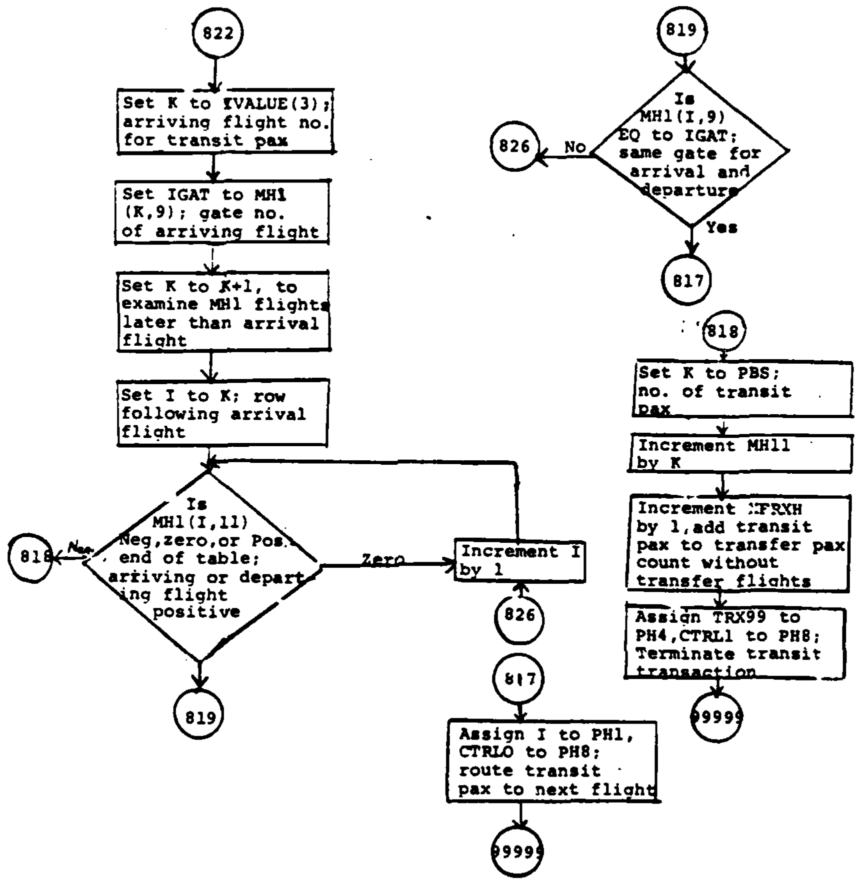

728
Set LOTNO EQ. PB14; parking lot number
Is LOTNO. EQ 0; is parking area assigned
Yes
Set LOTNO. EQ 1; general parking area
722
Set I to INDEXF(11); index number for rental cars
Set J to I+NORENT; last MH9 row no. for rental cars
Increment I by 1; first MH9 row no. for rental cars
Set N to I
Is MH9(N,4) NE IV5; Is passenger acengy code not same as rental agency number N
Yes
NO
Set L to MH9(N,5); parking facility no. for rental agency
Is L GT 1; is it a special lot
Yes
NO
723
Set N to INDEXF (10)+L; MH9 row number for parking facility number L
Set M to PARQS +L-1; queue-storage for parking facility number L
Increment N by 1; next MH9 row no. for rental agency
Is N GT J; all rental agencies been tried
NO
725
yes
728
HH
Is IV6 NE. 1; flag for area no. only
Yes
No
Set PB14 to LOTNO; area number
9999
9999
Set PB14 to LOTNO; parking lot number
Set PH11 to 10; process code for parking
Set PH7 to N; MH9 row no. of parking facility
Set PH5 to M; queue-storage no. of parking facility
Set PH2 to value of NPTTO; point no. of parking facility
727
Set N to INDEXF(10) + LOTNO; MH9 row number for parking facility
Set M to PARQS + LOTNO-1 queue-storage no. for parking facility LOTNO
HH
724
Set NPTTO to MH9 (N,3), point no. of parking facility
Is NPTFM EQ 0 Is it first Landside Facility
YES
727
NO
Assign statement number 727 to NEXT
950

# TRANSFER PAX

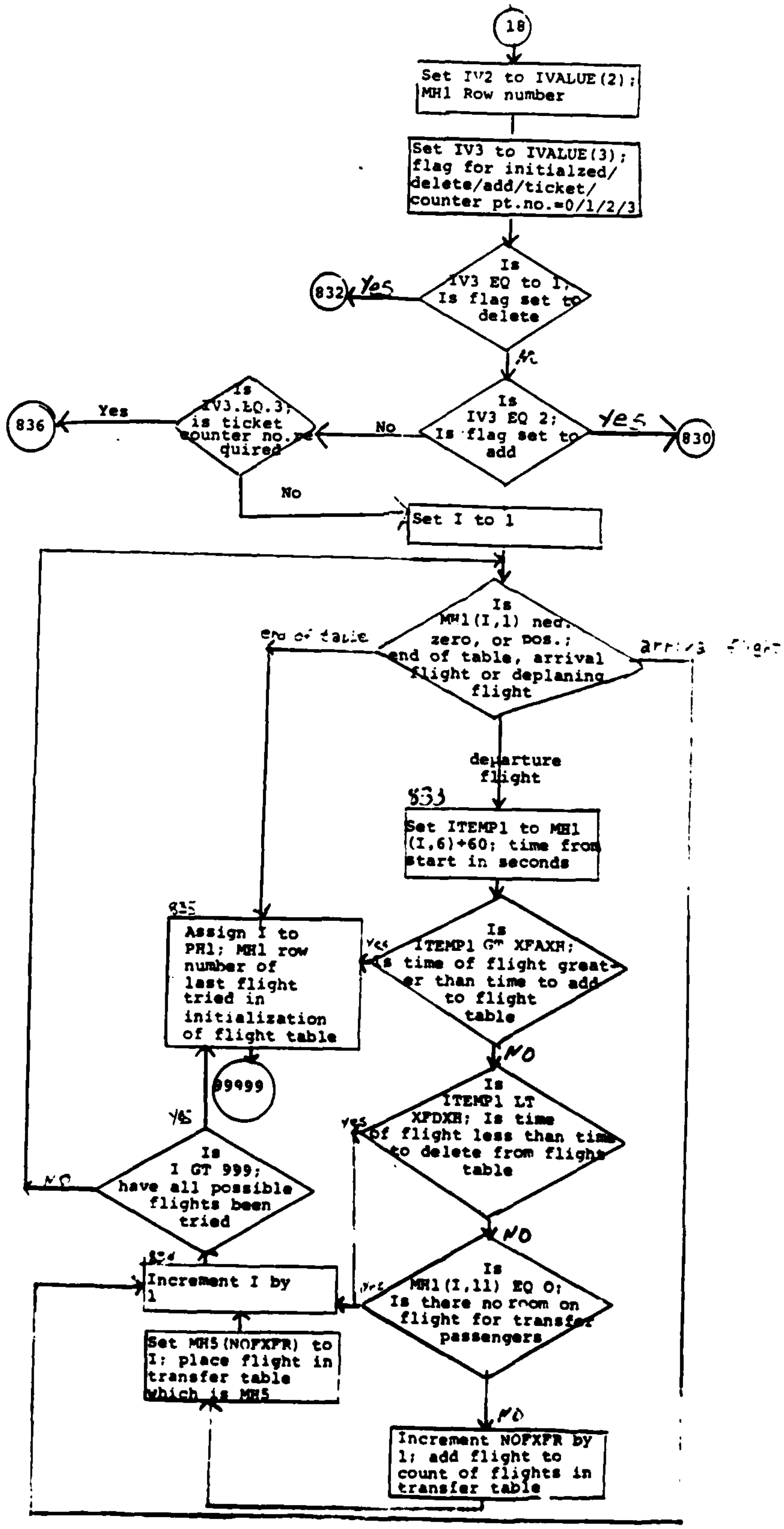


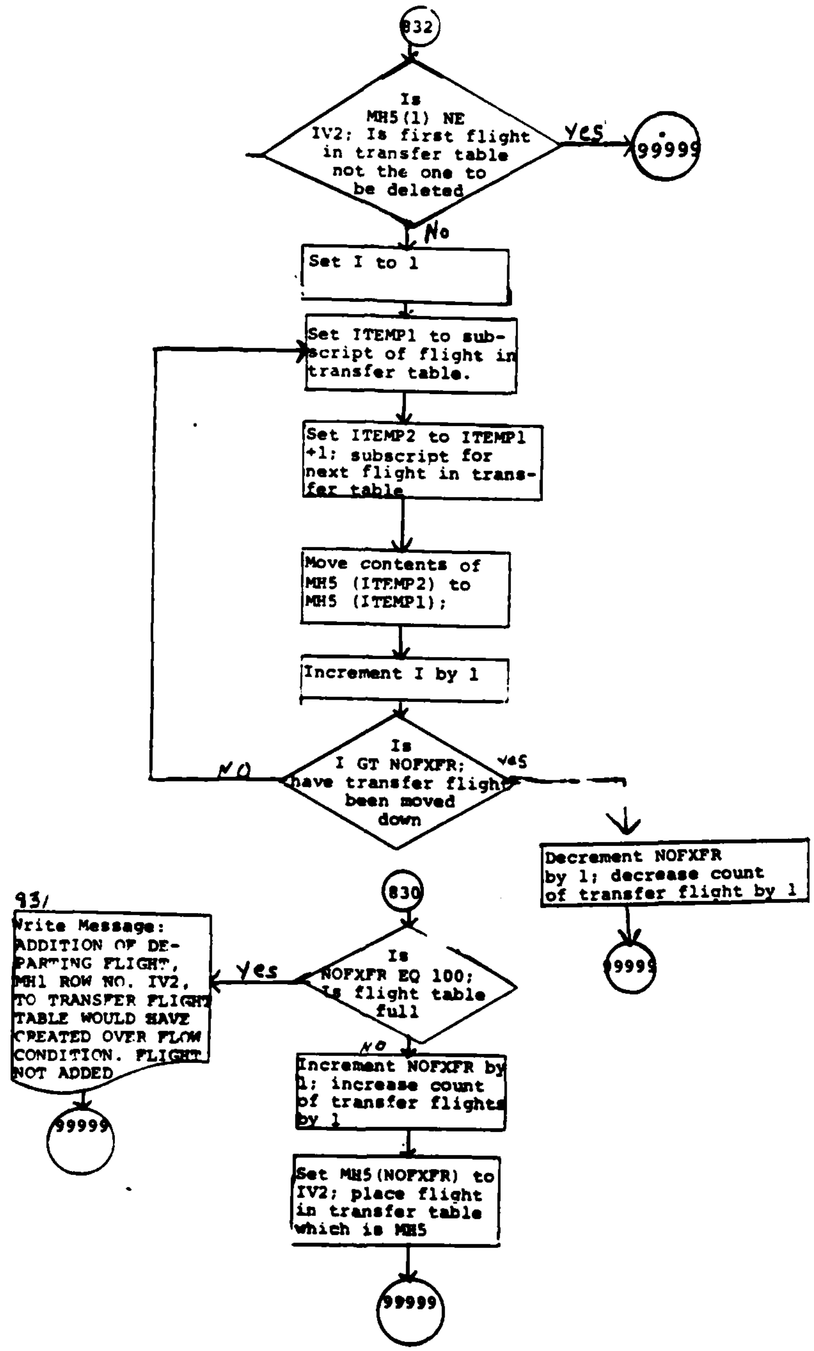


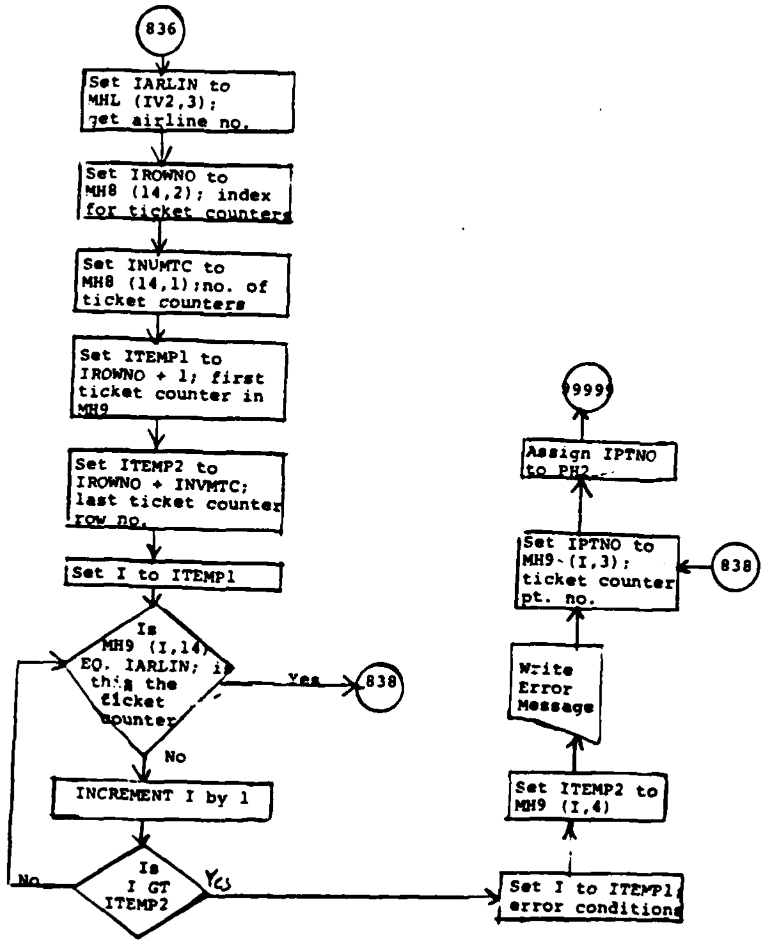
I
Is
MH1(I,11) GT 0
has space for transfers
been filled on
flight
No
Yes
820
Assign I to Ph1;
transfer flight
row in MH1
822
Set L to N; no.
in MH5 of flight
just filled
Set ITEMP3 to
MH5(L); sub-
script of trans-
fer flight in
MH5
Set ITEMP2 to
ITEMP3+1; sub-
script of next
flight in MH5
Set MH5(ITEMP3)
to MH5(ITEMP3);
move flight down
in MH5 matrix
savevalue
Increment L by 1
823
Is
L GT NOFXFR;
have all flights
been moved
down
No
yes
Decrement NOFXFR by
1; decrement no. of
transfer flights
available by one
820


822
Set K to *VALUE(3); arriving flight no. for transit pax
Set IGAT to MH1 (K,9); gate no. of arriving flight
Set K to K+1, to examine MH1 flights later than arrival flight
Set I to K; row following arrival flight
Is MH1(I,11) Neg,zero,or Pos. end of table; arriving or departing flight positive
818
Neg.
Zero
Increment I by 1
826
819
817
Assign I to PH1, CTRLO to PH8; route transit pax to next flight
9999
819
Is MH1(I,9) EQ to IGAT; same gate for arrival and departure
No
826
Yes
817
818
Set K to PBS; no. of transit pax
Increment MH11 by K
Increment XFRXH by 1,add transit pax to transfer pax count without transfer flights
Assign TRX99 to PH4,CTRL1 to PH8; Terminate transit transaction
9999

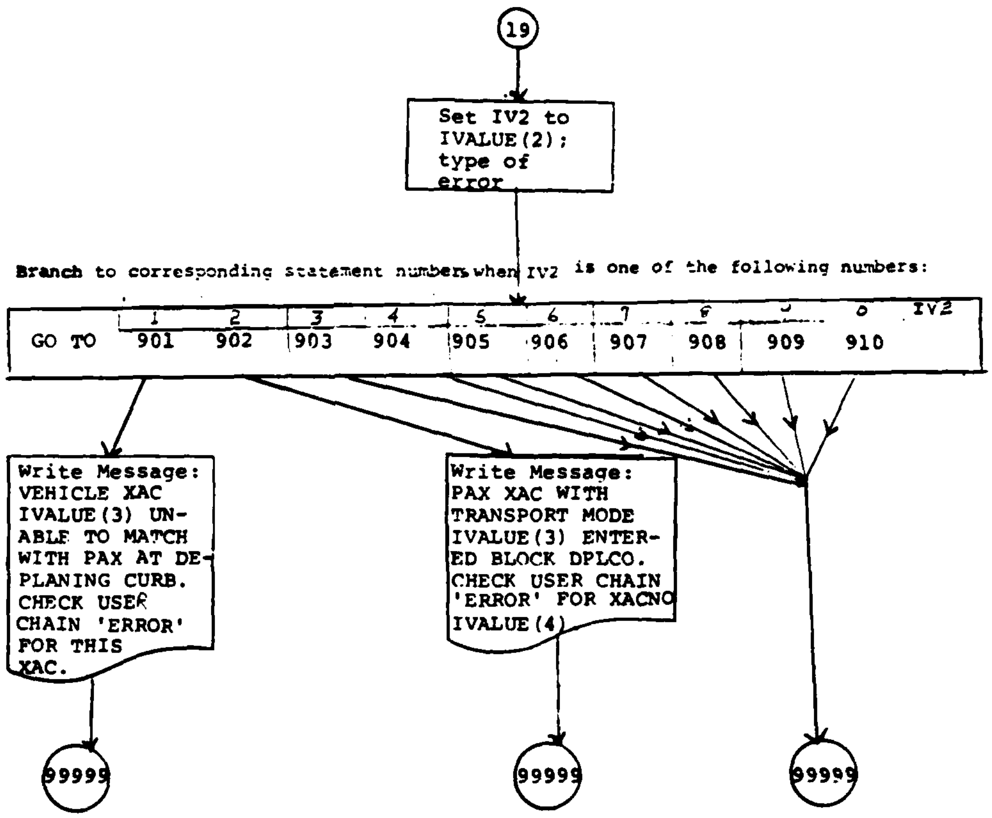
TRANSFER FLIGHTS



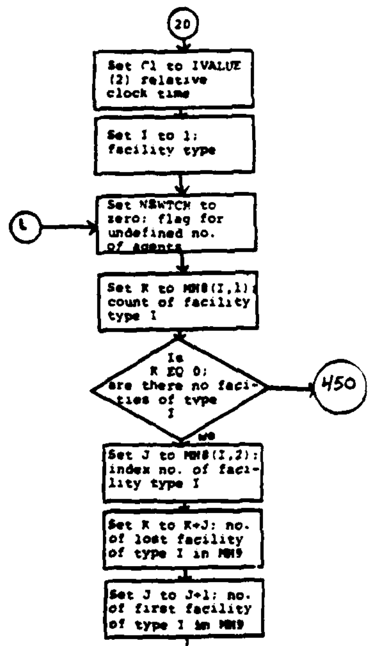

832
Is MH5(1) NE IV2; Is first flight in transfer table not the one to be deleted
Yes
99999
No
Set I to 1
Set ITEMP1 to subscript of flight in transfer table.
Set ITEMP2 to ITEMP1 +1; subscript for next flight in transfer table
Move contents of MH5 (ITEMP2) to MH5 (ITEMP1);
Increment I by 1
Is I GT NOFXFR; have transfer flights been moved down
NO
yes
Decrement NOFXFR by 1; decrease count of transfer flight by 1
99999
830
Is NOFXFR EQ 100; Is flight table full
Yes
931
Write Message: ADDITION OF DEPARTING FLIGHT, MH1 ROW NO. IV2, TO TRANSFER FLIGHT TABLE WOULD HAVE CREATED OVER FLOW CONDITION. FLIGHT NOT ADDED
99999
NO
Increment NOFXFR by 1; increase count of transfer flights by 1
Set MH5(NOFXFR) to IV2; place flight in transfer table which is MH5
99999


836
Set IARLIN to MHL (IV2,3); get airline no.
Set IROWNO to MH8 (14,2); index for ticket counters
Set INUMTC to MH8 (14,1); no. of ticket counters
Set ITEMP1 to IROWNO + 1; first ticket counter in MH9
Set ITEMP2 to IROWNO + INVMTC; last ticket counter row no.
Set I to ITEMP1
Is MH9 (I,14) EQ. IARLIN; is this the ticket counter
Yes
838
No
INCREMENT I by 1
Is I GT ITEMP2
No
Yes
Set I to ITEMP1; error condition
Set ITEMP2 to MH9 (I,4)
Write Error Message
Set IPTNO to MH9 (I,3); ticket counter pt. no.
838
Assign IPTNO to PH2
9999

## MISCELLANEOUS ERROR CONDITIONS



## FORMATED REPORTS




Branch to corresponding statement number when I is one of the following numbers:

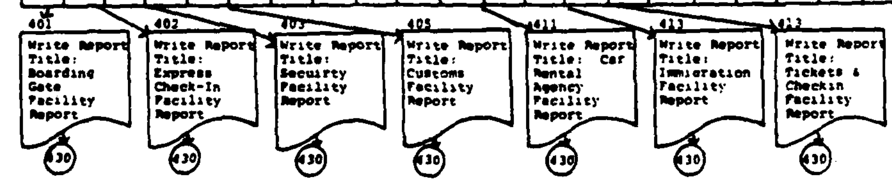
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 400 | 400 | 400 | 450 | 400 | 450 | 450 | 450 | 450 | 450 | 400 | 450 | 400 | 400 | 450 | 450 | 450 | 450 | 450 | 450 | St. No. |

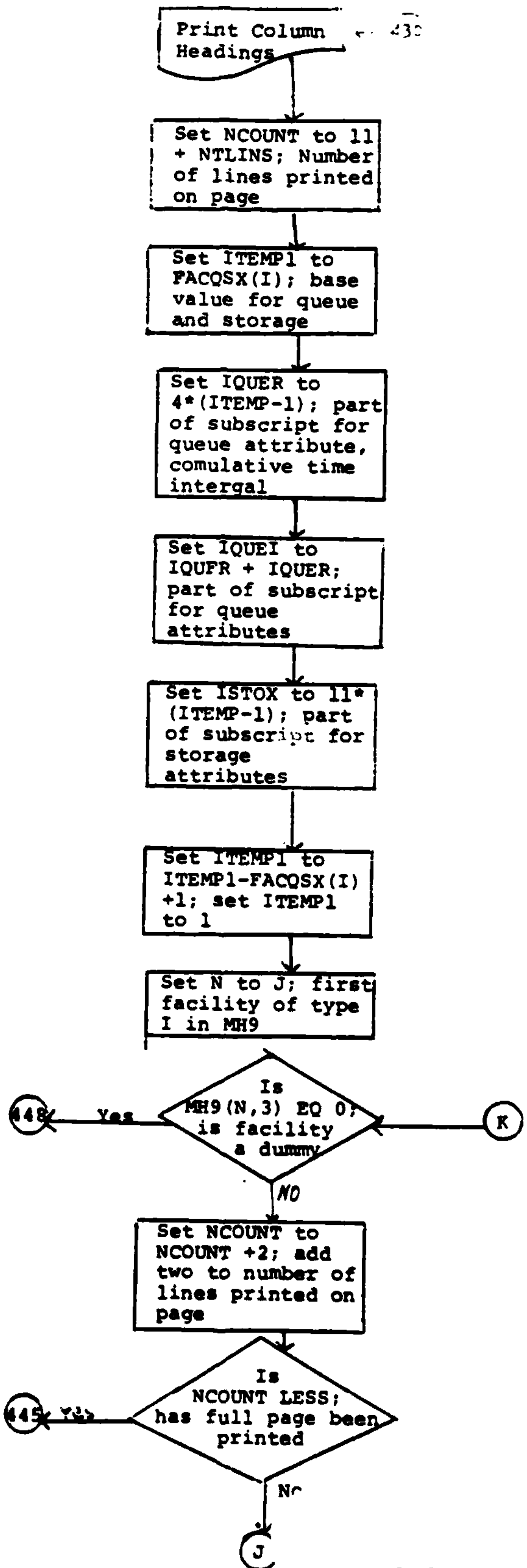
Continue to make reports for gates, customs, security, express checkin, ticketing, car rental, and immigration, skip other facility types.

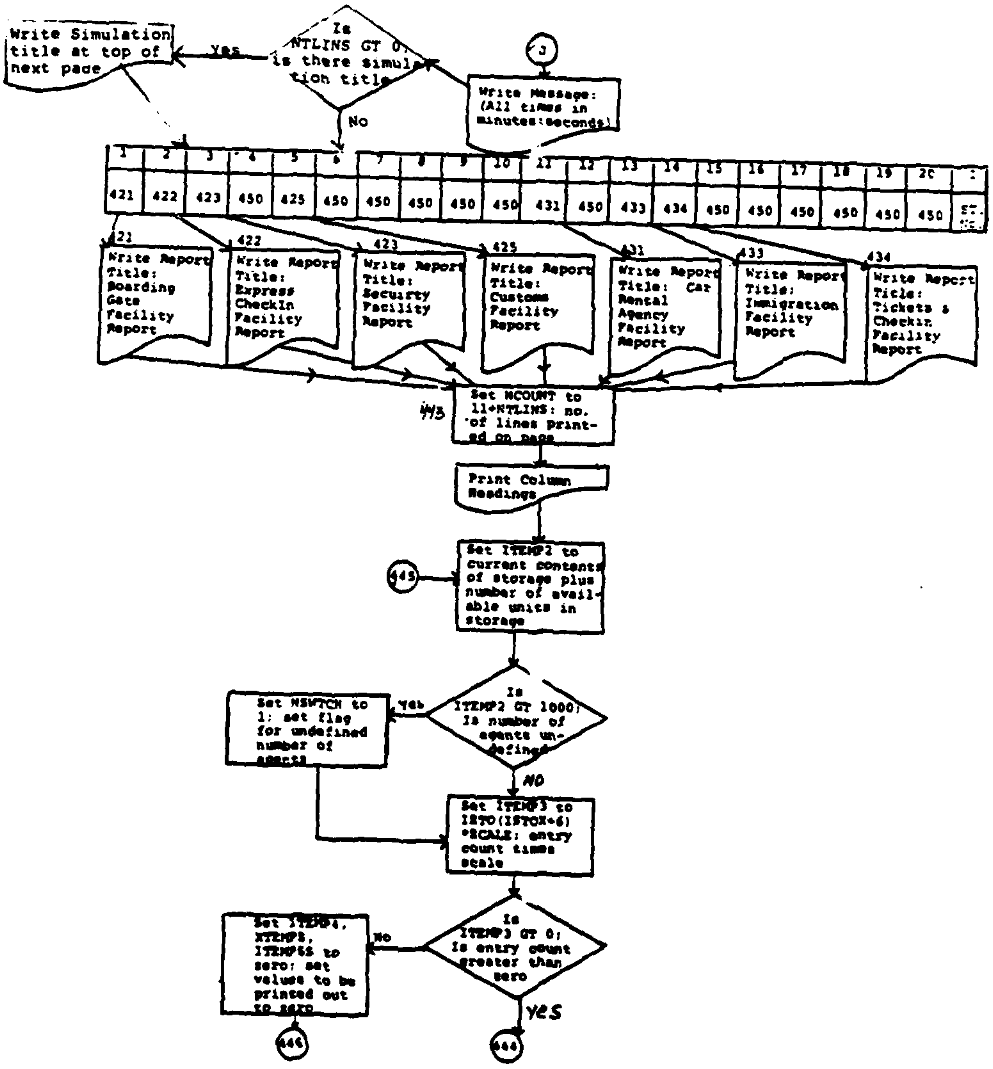
400

Branch to corresponding statement number when I is one of the following numbers:

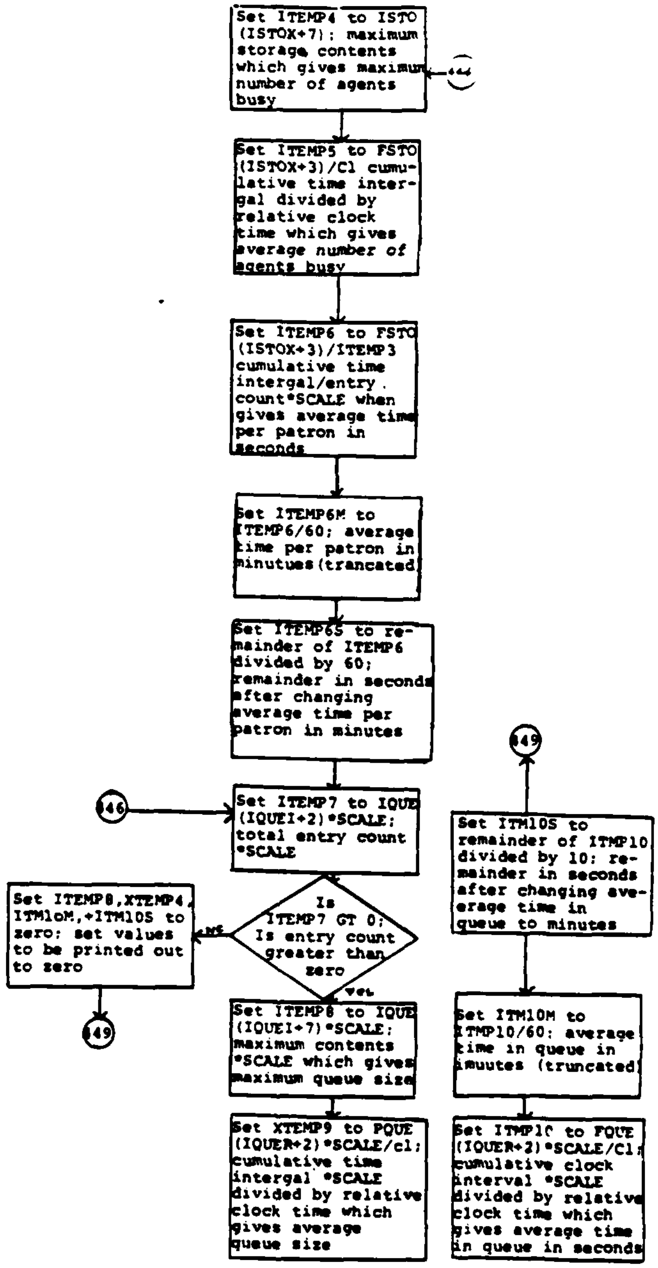
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 401 | 402 | 403 | 450 | 405 | 450 | 450 | 450 | 450 | 450 | 411 | 450 | 413 | 414 | 450 | 450 | 450 | 450 | 450 | 450 | St. No. |

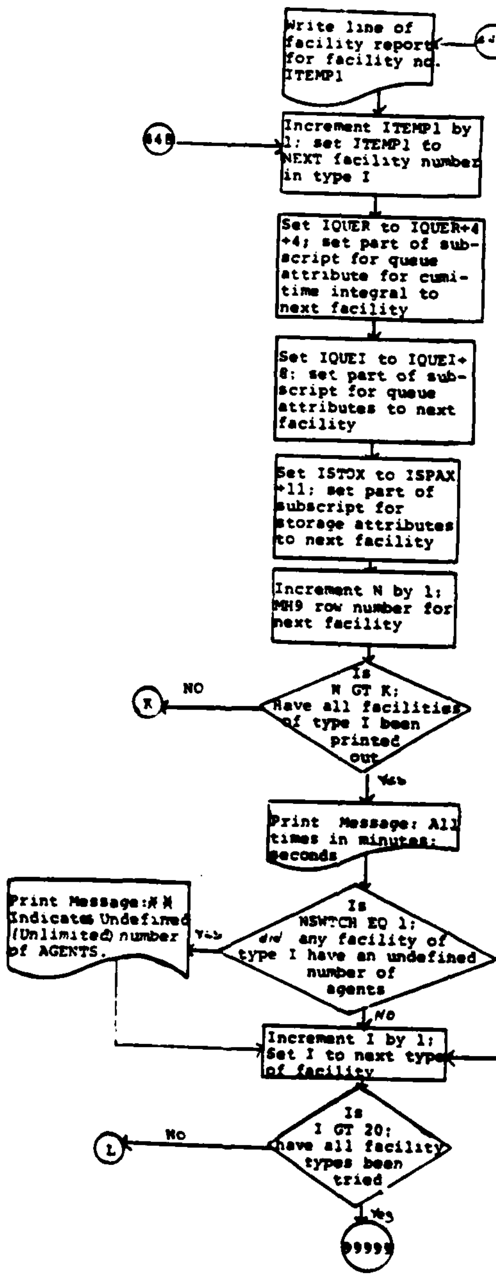



Print Column Headings
432
Set NCOUNT to 11 + NTLINS; Number of lines printed on page
Set ITEMP1 to FACQSX(I); base value for queue and storage
Set IQUER to 4*(ITEMP-1); part of subscript for queue attribute, comulative time intergal
Set IQUEI to IQUFR + IQUER; part of subscript for queue attributes
Set ISTOX to 11*(ITEMP-1); part of subscript for storage attributes
Set ITEMP1 to ITEMP1-FACQSX(I) +1; set ITEMP1 to 1
Set N to J; first facility of type I in MH9
Is MH9(N,3) EQ 0; is facility a dummy
Yes
448
K
NO
Set NCOUNT to NCOUNT +2; add two to number of lines printed on page
Is NCOUNT LESS; has full page been printed
445
No
J

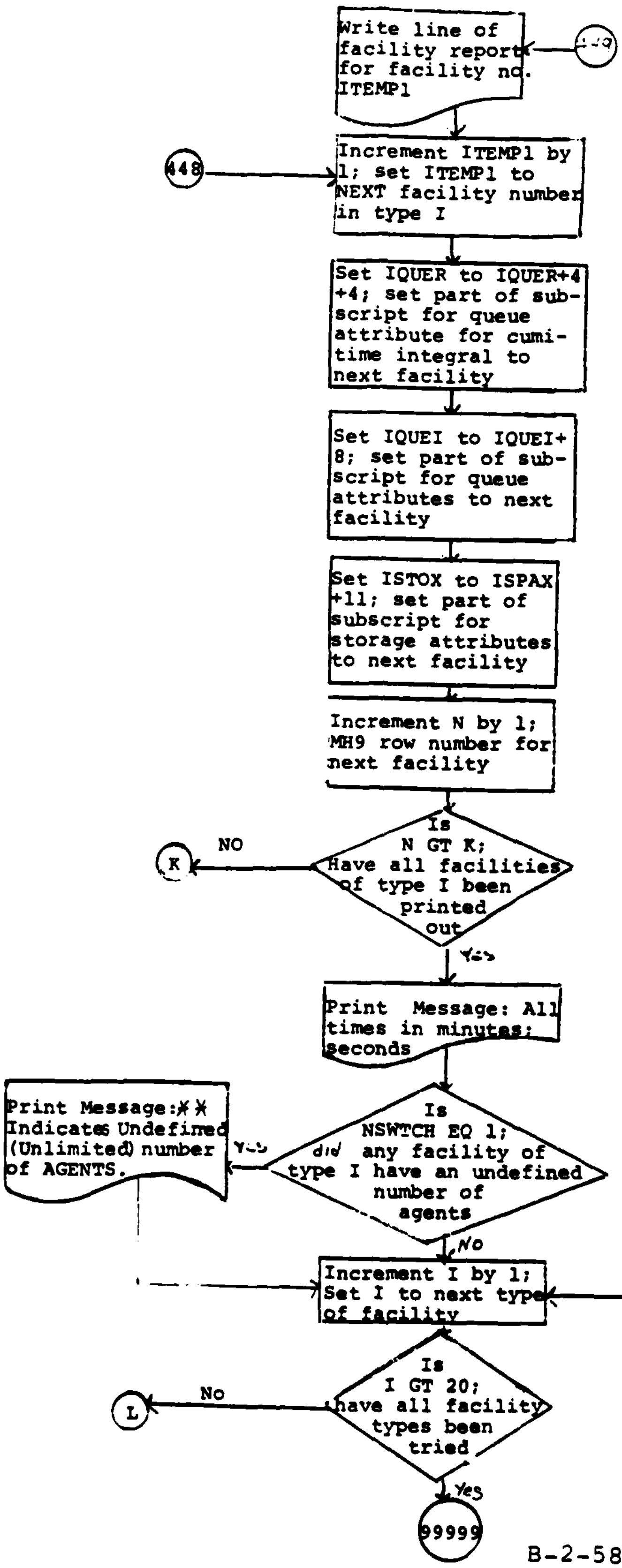


Write Simulation title at top of next page
Yes
Is NTLINS GT 0; is there simulation title
No
J
Write Message: (All times in minutes:seconds)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
421 422 423 450 425 450 450 450 450 450 431 450 433 434 450 450 450 450 450 450
421
Write Report Title: Boarding Gate Facility Report
422
Write Report Title: Express CheckIn Facility Report
423
Write Report Title: Security Facility Report
425
Write Report Title: Customs Facility Report
431
Write Report Title: Car Rental Agency Facility Report
433
Write Report Title: Immigration Facility Report
434
Write Report Title: Tickets & Checkin Facility Report
443
Set NCOUNT to 11+NTLINS: no. of lines printed on page
Print Column Headings
445
Set ITEMP2 to current contents of storage plus number of available units in storage
Is ITEMP2 GT 1000; Is number of agents undefined
Yes
Set NSWTCH to 1: set flag for undefined number of agents
NO
Is ITEMP3 GT 0; Is entry count greater than zero
No
YES
446
444

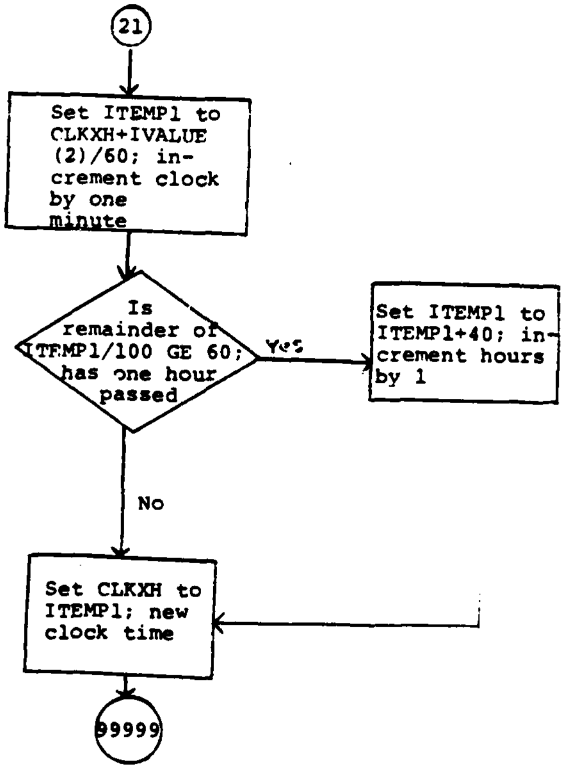

Set ITEMP4 to ISTO (ISTOX+7); maximum storage contents which gives maximum number of agents busy
444
Set ITEMP5 to FSTO (ISTOX+3)/C1 cumulative time intergal divided by relative clock time which gives average number of agents busy
Set ITEMP6 to FSTO (ISTOX+3)/ITEMP3 cumulative time intergal/entry count*SCALE when gives average time per patron in seconds
Set ITEMP6M to ITEMP6/60; average time per patron in minutues (truncated)
Set ITEMP6S to remainder of ITEMP6 divided by 60; remainder in seconds after changing average time per patron in minutes
446
Set ITEMP7 to IQUE (IQUEI+2)*SCALE: total entry count *SCALE
Is ITEMP7 GT 0; Is entry count greater than zero
NO
YES
Set ITEMP8,XTEMP4, ITM1OM,+ITM10S to zero; set values to be printed out to zero
449
Set ITEMP8 to IQUE (IQUEI+7)*SCALE: maximum contents *SCALE which gives maximum queue size
Set XTEMP9 to FQUE (IQUER+2)*SCALE/cl; cumulative time intergal *SCALE divided by relative clock time which gives average queue size
449
Set ITM10S to remainder of ITMP10 divided by 10: remainder in seconds after changing average time in queue to minutes
Set ITM10M to ITMP10/60: average time in queue in imuutes (truncated)
Set ITMP10 to FQUE (IQUER+2)*SCALE/C1; cumulative clock interval *SCALE divided by relative clock time which gives average time in queue in seconds



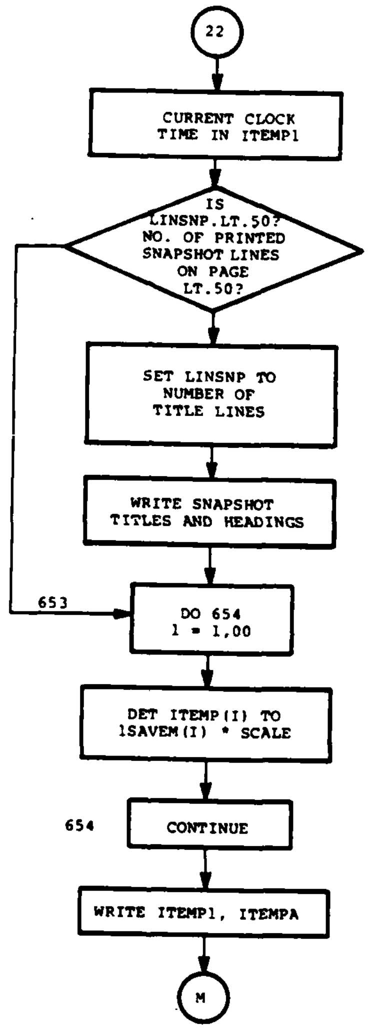
Write line of facility report for facility no. ITEMP1
449
448
Increment ITEMP1 by 1: set ITEMP1 to NEXT facility number in type I
Set IQUER to IQUER+4 +4; set part of subscript for queue attribute for cumitime integral to next facility
Set IQUEI to IQUEI+8: set part of subscript for queue attributes to next facility
Set ISTOX to ISPAX +11: set part of subscript for storage attributes to next facility
Increment N by 1: MH9 row number for next facility
Is N GT K: Have all facilities of type I been printed out
NO
K
Yes
Print Message: All times in minutes: seconds
Is NSWTCH EQ 1: did any facility of type I have an undefined number of agents
Yes
Print Message: N N Indicates Undefined (Unlimited) number of AGENTS.
NO
Increment I by 1: Set I to next type of facility
450
Is I GT 20: have all facility types been tried
No
L
Yes
9999

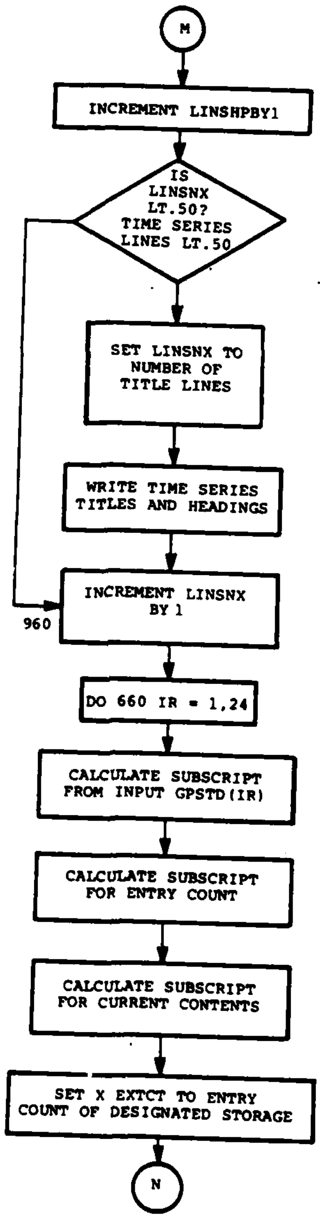

Write line of facility report for facility no. ITEMP1
448
Increment ITEMP1 by 1; set ITEMP1 to NEXT facility number in type I
Set IQUER to IQUER+4 +4; set part of sub- script for queue attribute for cumi- time integral to next facility
Set IQUEI to IQUEI+ 8; set part of sub- script for queue attributes to next facility
Set ISTOX to ISPAX +11; set part of subscript for storage attributes to next facility
Increment N by 1; MH9 row number for next facility
Is N GT K; Have all facilities of type I been printed out
NO
K
Yes
Print Message: All times in minutes; seconds
Is NSWTCH EQ 1; did any facility of type I have an undefined number of agents
Yes
Print Message:** Indicates Undefined (Unlimited) number of AGENTS.
No
Increment I by 1; Set I to next type of facility
450
Is I GT 20; have all facility types been tried
No
L
Yes
9999

CLOCK UPDATE

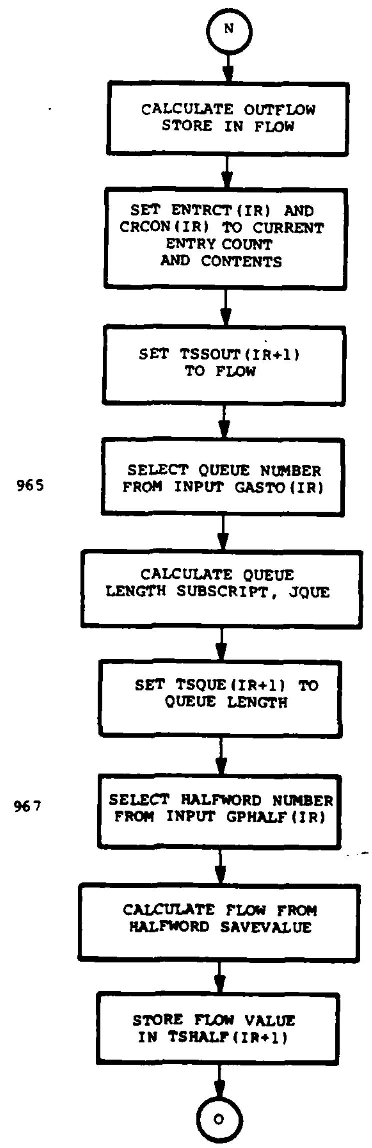


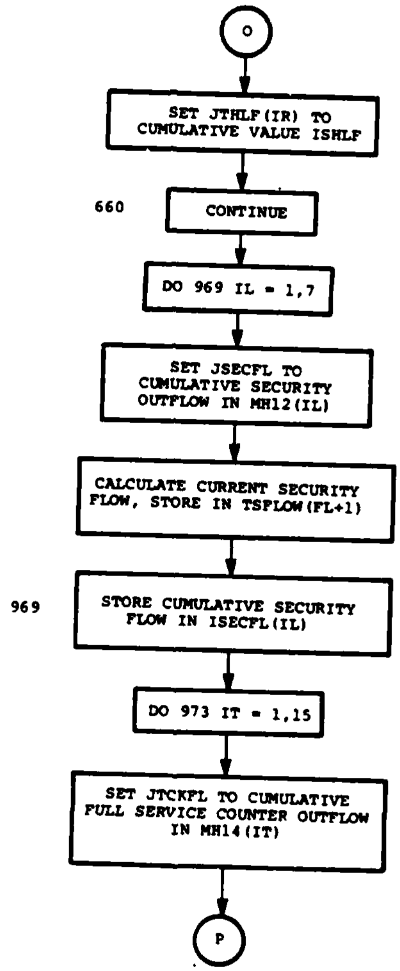
SNAPSHOTS

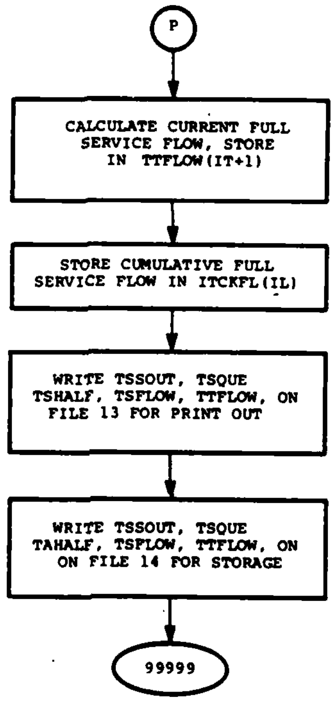

22
CURRENT CLOCK TIME IN ITEMP1
IS LINSNP.LT.50? NO. OF PRINTED SNAPSHOT LINES ON PAGE LT.50?
SET LINSNP TO NUMBER OF TITLE LINES
WRITE SNAPSHOT TITLES AND HEADINGS
653
DO 654 1 = 1,00
DET ITEMP(I) TO 1SAVEM(I) * SCALE
654
CONTINUE
WRITE ITEMP1, ITEMPA
M


M
INCREMENT LINSHPBY1
IS LINSNX LT.50? TIME SERIES LINES LT.50
SET LINSNX TO NUMBER OF TITLE LINES
WRITE TIME SERIES TITLES AND HEADINGS
INCREMENT LINSNX BY 1
960
DO 660 IR = 1,24
CALCULATE SUBSCRIPT FROM INPUT GPSTD(IR)
CALCULATE SUBSCRIPT FOR ENTRY COUNT
CALCULATE SUBSCRIPT FOR CURRENT CONTENTS
SET X EXTCT TO ENTRY COUNT OF DESIGNATED STORAGE
N

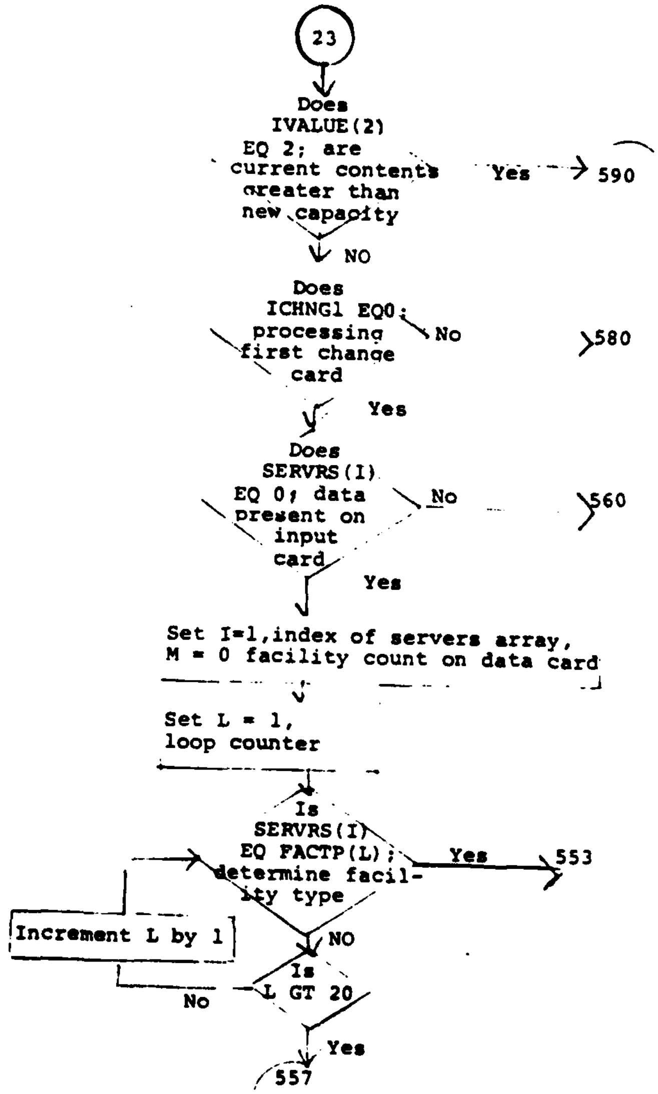
B

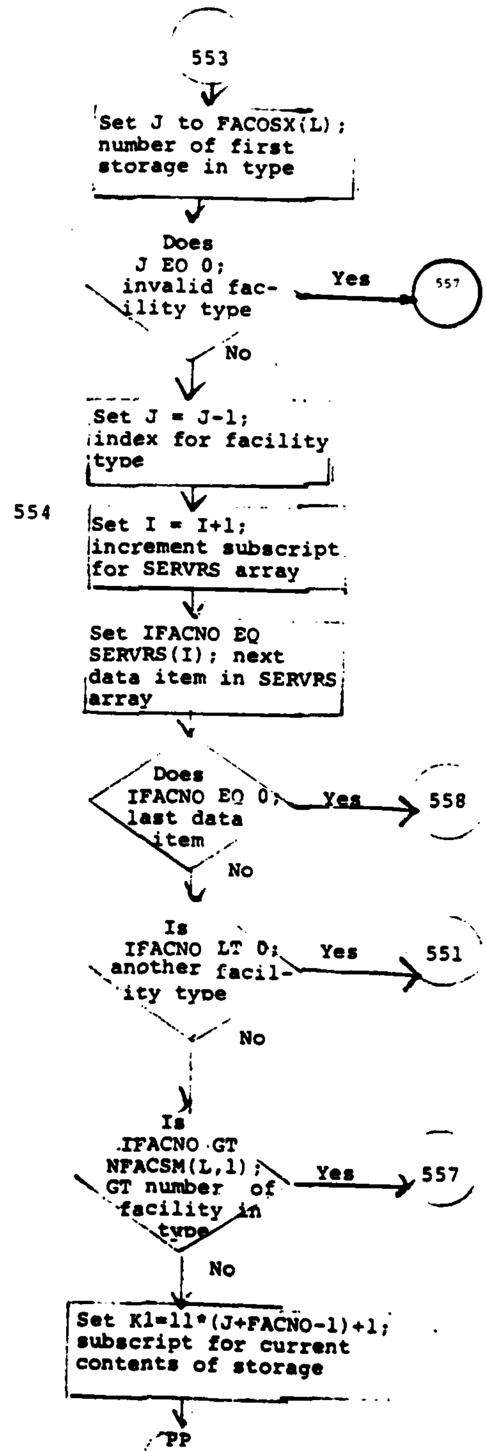

N
CALCULATE OUTFLOW STORE IN FLOW
SET ENTRCT(IR) AND CRCON(IR) TO CURRENT ENTRY COUNT AND CONTENTS
SET TSSOUT(IR+1) TO FLOW
965
SELECT QUEUE NUMBER FROM INPUT GASTO(IR)
CALCULATE QUEUE LENGTH SUBSCRIPT, JQUE
SET TSQUE(IR+1) TO QUEUE LENGTH
967
SELECT HALFWORD NUMBER FROM INPUT GPHALF(IR)
CALCULATE FLOW FROM HALFWORD SAVEVALUE
STORE FLOW VALUE IN TSHALF(IR+1)
O

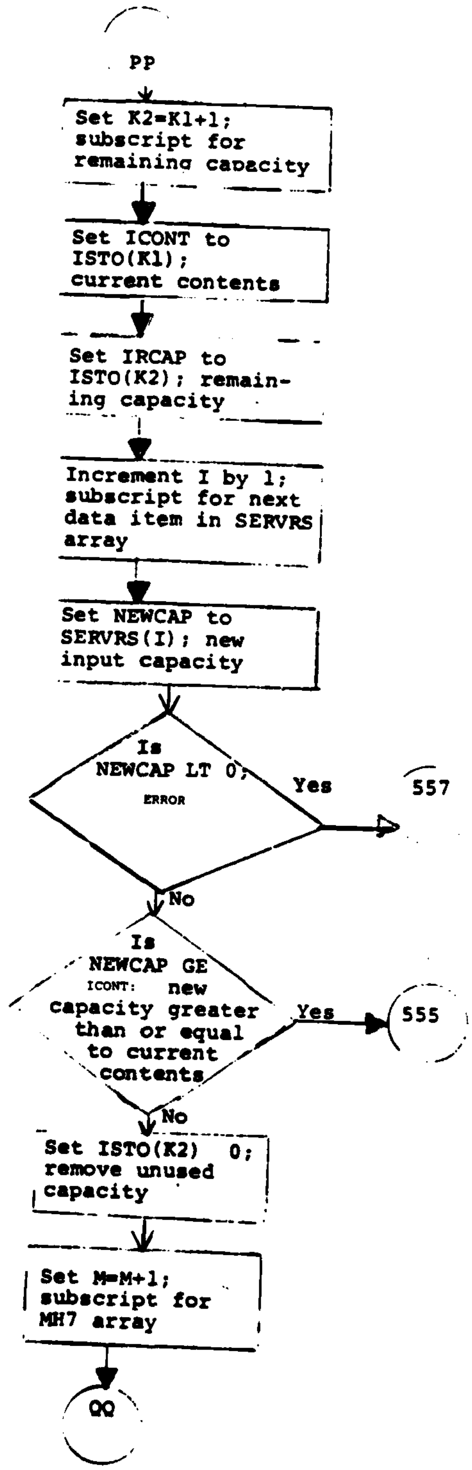

O
SET JTHLF(IR) TO CUMULATIVE VALUE ISHLF
660
CONTINUE
DO 969 IL = 1,7
SET JSECFL TO CUMULATIVE SECURITY OUTFLOW IN MH12(IL)
CALCULATE CURRENT SECURITY FLOW, STORE IN TSFLOW(FL+1)
969
STORE CUMULATIVE SECURITY FLOW IN ISECFL(IL)
DO 973 IT = 1,15
SET JTCKFL TO CUMULATIVE FULL SERVICE COUNTER OUTFLOW IN MH14(IT)
P

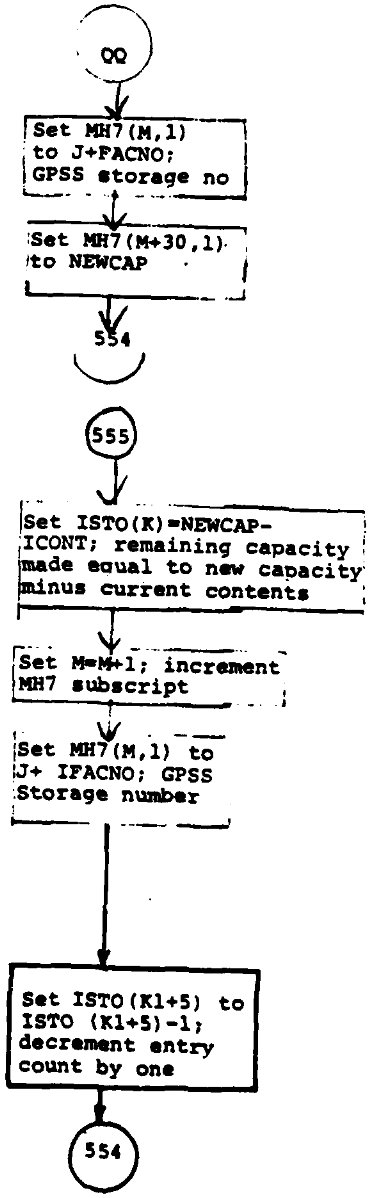

P
CALCULATE CURRENT FULL SERVICE FLOW, STORE IN TTFLOW(IT+1)
STORE CUMULATIVE FULL SERVICE FLOW IN ITCKFL(IL)
WRITE TSSOUT, TSQUE TSHALF, TSFLOW, TTFLOW, ON FILE 13 FOR PRINT OUT
WRITE TSSOUT, TSQUE TAHALF, TSFLOW, TTFLOW, ON ON FILE 14 FOR STORAGE
99999

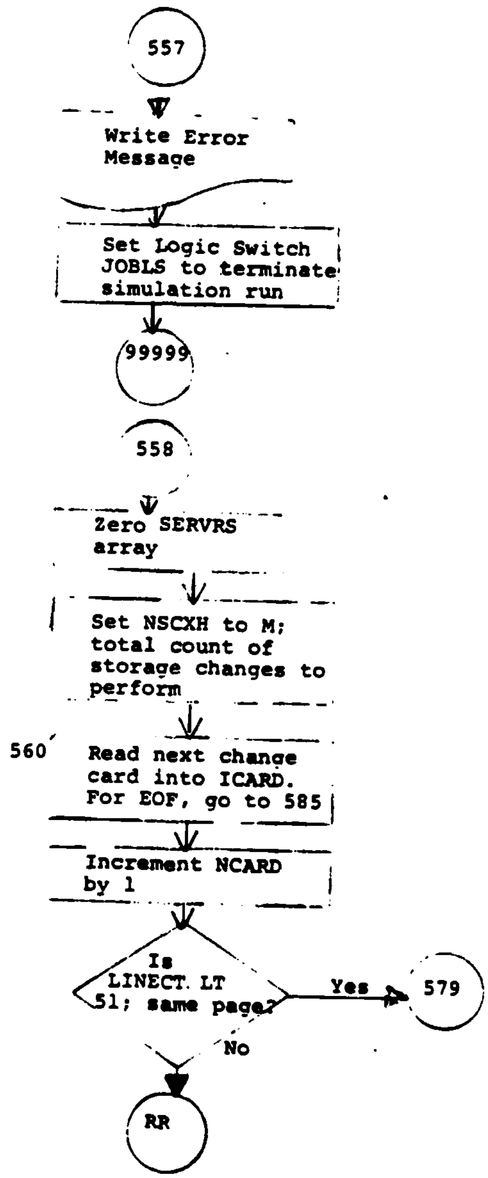
## CHANGE CARD PROCESSING

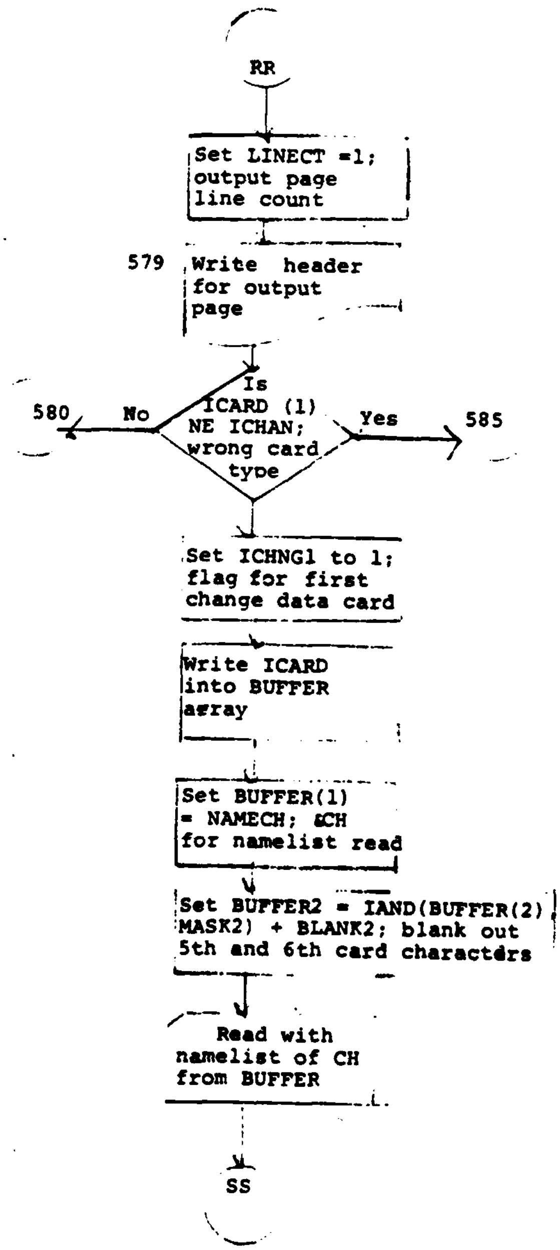



553
Set J to FACOSX(L); number of first storage in type
Does J EO 0; invalid facility type
Yes
557
No
Set J = J-1; index for facility type
554
Set I = I+1; increment subscript for SERVRS array
Set IFACNO EQ SERVRS(I); next data item in SERVRS array
Does IFACNO EQ 0; last data item
Yes
558
No
Is IFACNO LT 0; another facility type
Yes
551
No
Is IFACNO GT NFACSM(L,1); GT number of facility in type
Yes
557
No
Set K1=11*(J+FACNO-1)+1; subscript for current contents of storage
PP


PP
Set K2=K1+1; subscript for remaining capacity
Set ICONT to ISTO(K1); current contents
Set IRCAP to ISTO(K2); remaining capacity
Increment I by 1; subscript for next data item in SERVRS array
Set NEWCAP to SERVRS(I); new input capacity
Is NEWCAP LT 0; ERROR
Yes
557
No
Is NEWCAP GE ICONT: new capacity greater than or equal to current contents
Yes
555
No
Set ISTO(K2) 0; remove unused capacity
Set M=M+1; subscript for MH7 array
QQ


QQ
Set MH7(M,1) to J+FACNO; GPSS storage no
Set MH7(M+30,1) to NEWCAP
554
555
Set ISTO(K)=NEWCAP-ICONT; remaining capacity made equal to new capacity minus current contents
Set M=M+1; increment MH7 subscript
Set MH7(M,1) to J+ IFACNO; GPSS Storage number
Set ISTO(K1+5) to ISTO (K1+5)-1; decrement entry count by one
554


557
Write Error Message
Set Logic Switch JOBLS to terminate simulation run
99999
558
Zero SERVRS array
Set NSCXH to M; total count of storage changes to perform
560
Read next change card into ICARD. For EOF, go to 585
Increment NCARD by 1
Is LINECT. LT 51; same page?
Yes
579
No
RR


RR
Set LINECT =1; output page line count
579
Write header for output page
Is ICARD (1) NE ICHAN; wrong card type
580
No
Yes
585
Set ICHNG1 to 1; flag for first change data card
Write ICARD into BUFFER array
Set BUFFER(1) = NAMECH; &CH for namelist read
Set BUFFER2 = IAND(BUFFER(2) MASK2) + BLANK2; blank out 5th and 6th card characters
Read with namelist of CH from BUFFER
SS

SS

Set IC= SAVEVALUE CLKXH; current time of 24 hour clock

Calculate seconds from current time to next storage change. Place in CHGXF

99999

585

Set CHGXF to 1,000,000 delay final change beyond simulation run time

99999

590

Set J=11*(IVALUE(3)-1) +1; current contents subscript

Set NEWCAP=IVALUE(4); GPSS value from MH7 (M,1)

TT

TT

Set NURCAP=NEWCAP
-ISTO(J); subtract
current contents
from new capacity

Is
NURCAP GE 0;
does capacity
exceed or equal
current
contents

592

Set ISTO(J+1)20;
set remaining
capacity to 0

99999

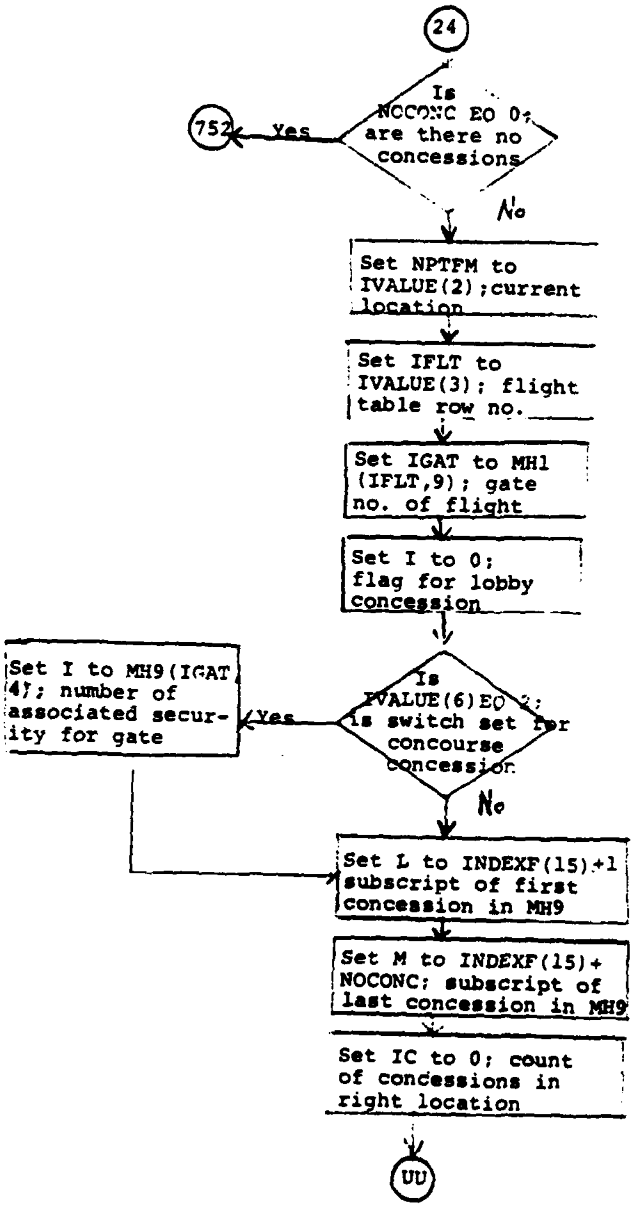
592

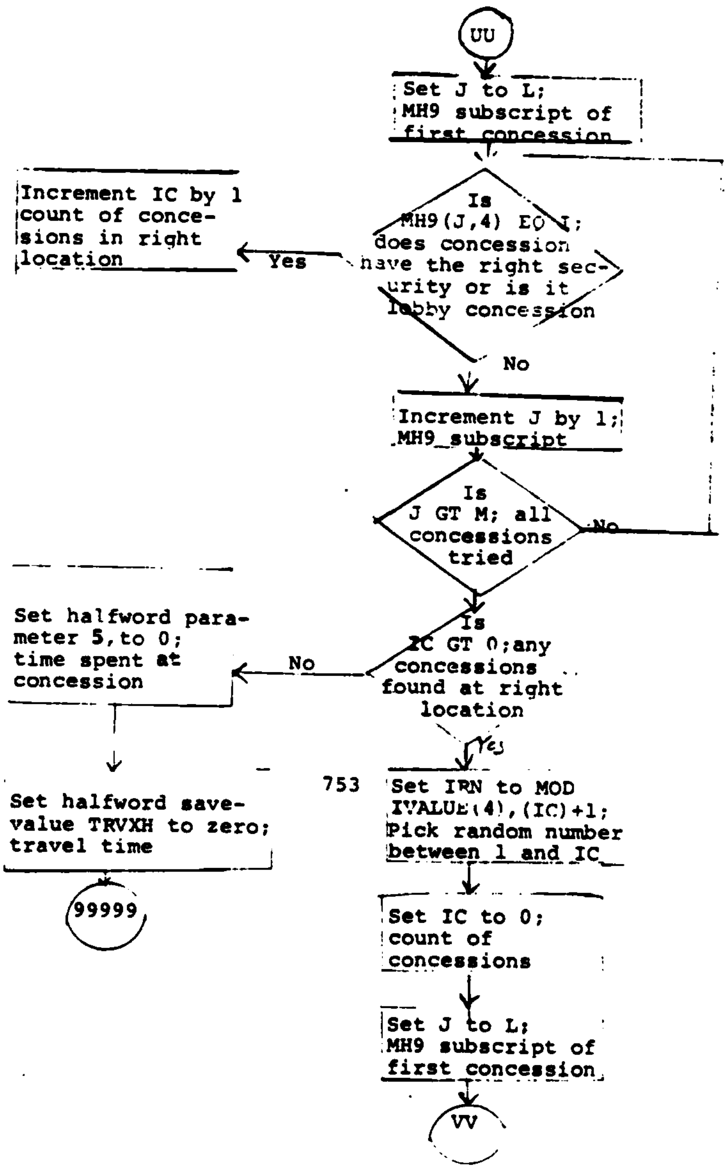
Set ISTO(J+1)=NURCAP;
remaining capacity
set to NURCAP

Set SCLXH to 1;
flag for storage
lowering complete

99999

# CONCESSION


24
Is NOCONC EQ 0; are there no concessions
Yes
752
No
Set NPTFM to IVALUE(2);current location
Set IFLT to IVALUE(3); flight table row no.
Set IGAT to MH1 (IFLT,9); gate no. of flight
Set I to 0; flag for lobby concession
Is IVALUE(6) EQ 2; is switch set for concourse concession
Yes
Set I to MH9(IGAT, 4); number of associated security for gate
No
Set L to INDEXF(15)+1 subscript of first concession in MH9
Set M to INDEXF(15)+ NOCONC; subscript of last concession in MH9
Set IC to 0; count of concessions in right location
UU


UU
Set J to L;
MH9 subscript of
first concession
Is
MH9(J,4) EQ I;
does concession
have the right sec-
urity or is it
lobby concession
Yes
Increment IC by 1
count of conce-
sions in right
location
No
Increment J by 1;
MH9 subscript
Is
J GT M; all
concessions
tried
No
Is
IC GT 0;any
concessions
found at right
location
No
Set halfword para-
meter 5,to 0;
time spent at
concession
Set halfword save-
value TRVXH to zero;
travel time
99999
Yes
753
Set IRN to MOD
IVALUE(4),(IC)+1;
Pick random number
between 1 and IC
Set IC to 0;
count of
concessions
Set J to L;
MH9 subscript of
first concession
VV

VV

Is
MH9(J,4) EQ I;
does concession
have the right
security or is
it lobby conces-
sion

Yes

Increment IC by 1;
count of concessions
at right location

No

Is
IC EQ IRN;
is number of
concession the
same as the
one ramdomly
picked

Yes

No

Increment J by 1;
MH9 subscript

Is
J GT M; all
concessions
tried

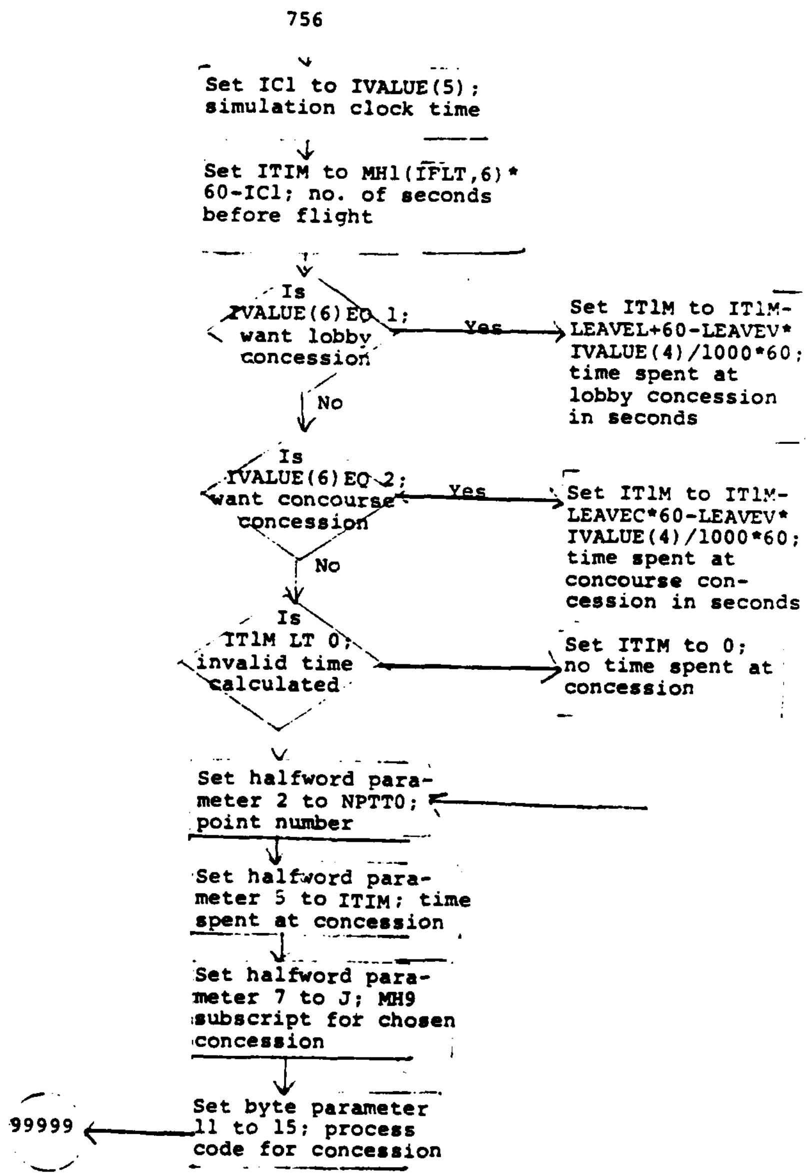
No

Yes

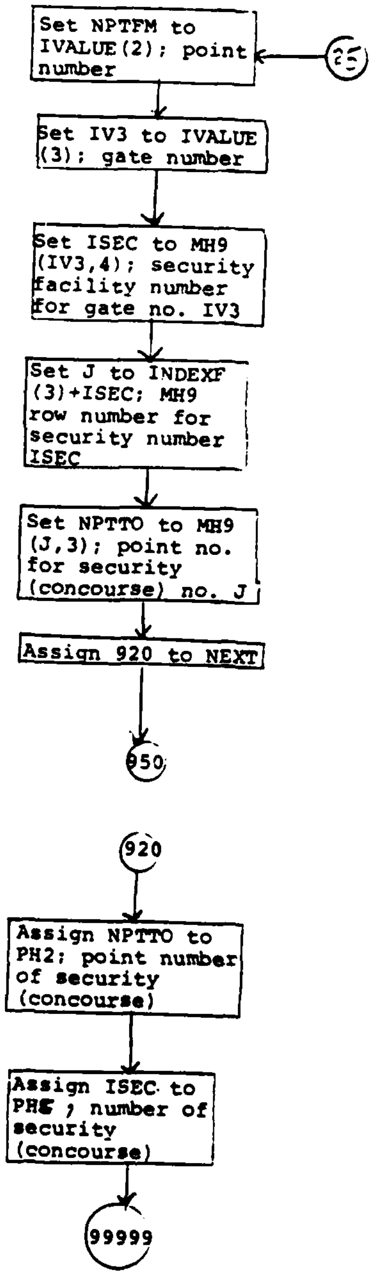
755 Set NTTO to MH9
(J,3); get pt. no.
of concessions

Assign 756 to NEXT;
return after walking
time defermination

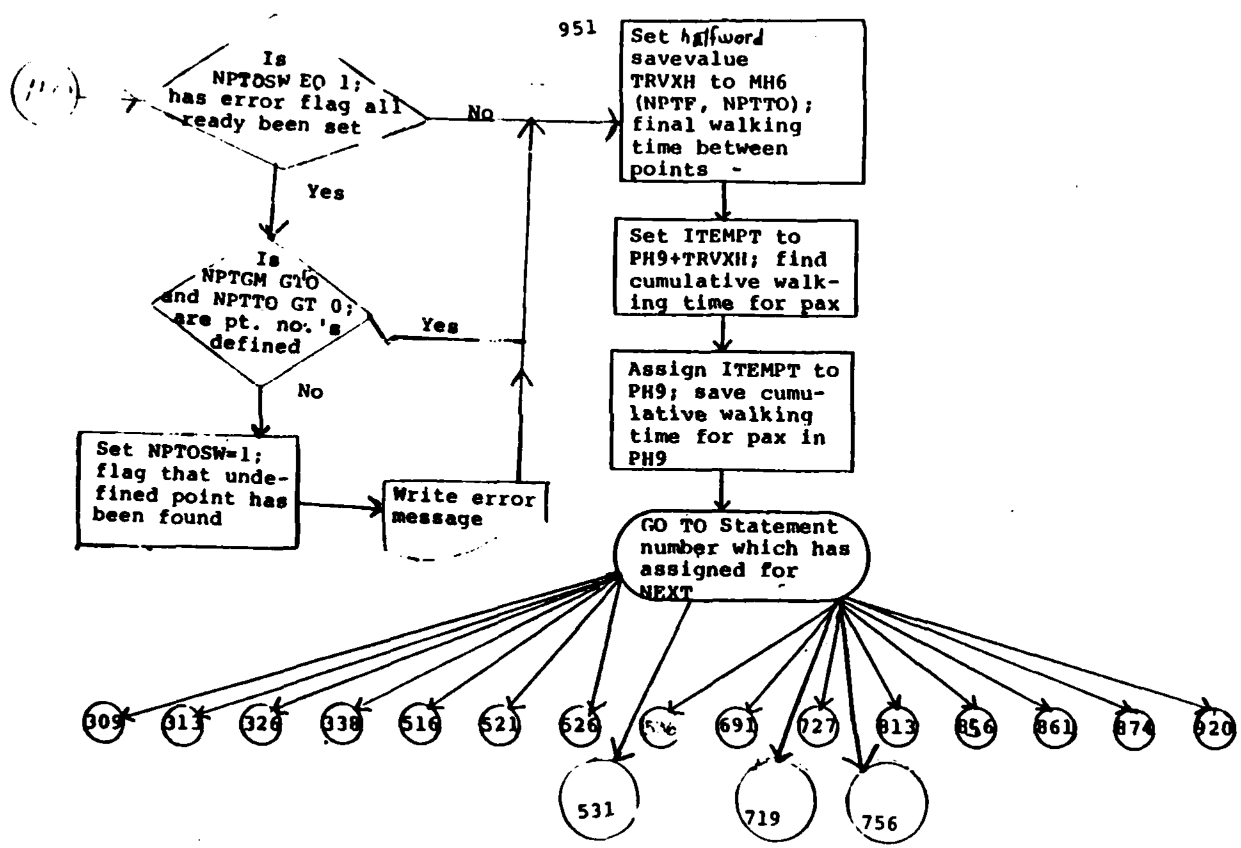
950


756
Set IC1 to IVALUE(5); simulation clock time
Set ITIM to MH1(IFLT,6)*60-IC1; no. of seconds before flight
Is IVALUE(6) EQ 1; want lobby concession
Yes
Set IT1M to IT1M-LEAVEL+60-LEAVEV*IVALUE(4)/1000*60; time spent at lobby concession in seconds
No
Is IVALUE(6) EQ 2; want concourse concession
Yes
Set IT1M to IT1M-LEAVEC*60-LEAVEV*IVALUE(4)/1000*60; time spent at concourse concession in seconds
No
Is IT1M LT 0; invalid time calculated
Set ITIM to 0; no time spent at concession
Set halfword parameter 2 to NPTT0; point number
Set halfword parameter 5 to ITIM; time spent at concession
Set halfword parameter 7 to J; MH9 subscript for chosen concession
Set byte parameter 11 to 15; process code for concession
99999

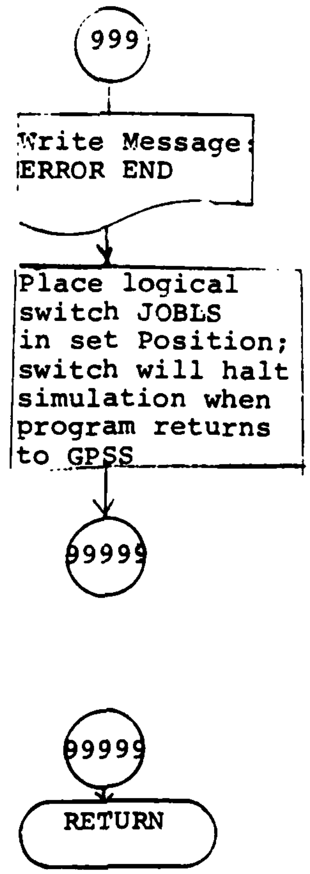
## CONCOURSE


Set NPTFM to IVALUE(2); point number
25
Set IV3 to IVALUE (3); gate number
Set ISEC to MH9 (IV3,4); security facility number for gate no. IV3
Set J to INDEXF (3)+ISEC; MH9 row number for security number ISEC
Set NPTTO to MH9 (J,3); point no. for security (concourse) no. J
Assign 920 to NEXT
950
920
Assign NPTTO to PH2; point number of security (concourse)
Assign ISEC to PH5 ; number of security (concourse)
99999

## WALKING TIME CALCULATION



ERROR ABEND


999
Write Message:
ERROR END
Place logical switch JOBLS in set Position; switch will halt simulation when program returns to GPSS
99999
99999
RETURN

APPENDIX B-3

LISTING OF FORTM SUBPROGRAM

```
C HELPC   LINKC --- HELPA   FORTM
C
C
      SUBROUTINE LINKC(IVALUE,ISAVEF,ISAVEH,IFAC,ISTO,FSTO,IQUE,
     *FQUE,ILOG,ITAB,FTAB,IUSE,IUSEF,FUSE,IMAX,IMAXB,IMAXH,IMAXBH,FSAVEL
     *,IMAXL,FMAXBL)
      REAL*8 FQUE,FUSE,FTAB
      INTEGER*2 ISAVEH,ILOG,IUSE,IMAXBH
      DIMENSION IVALUE(6),ISAVEF(2),ISAVEH(2),IFAC(2),ISTO(2),FSTO(2),
     *IQUE(2),FQUE(2),ILOG(2),ITAB(2),FTAB(2),IUSE(2),IUSEF(2),FUSE(2),
     *IMAX(2),IMAXB(2),IMAXH(2),IMAXBH(2),FSAVEL(2),IMAXL(2),FMAXBL(2)
C
      INTEGER PVAL
C
      REAL*8     DUM8,ZAP,NAMER8
      INTEGER*4 TIME,BUFFER,BLANK,BLANK1,FACTYP,TYPTST,START,FINISH
      INTEGER*4 BLANK2,SERVRS
      INTEGER*4 ERRORS,ASTRSK,FACQSX,FROMTO,FROM,TO
      INTEGER*4 ENDXF,TRVXH,BD1XH,ABUXH,DBUXH,XFRXH,XFAXH,XFDXH,SCLXH
      INTEGER*4 CLKXH,CHGXF,NSCXH,SLCXH,GRTXL,WWGXH,GRGXL,GRT00,CPKXH
       INTEGER*4 CGTXL, PCBXL, CRBXH, CONXH
      INTEGER*4 CUSOS,RCROS,CHKQS,DPCBS,EPCBS,SECQS,GAQSL,PARQS,TICQS
      INTEGER*4 RCARO,BAGCO,DPLCO,CHEK2,CHEK3,CGTRO,ERROR,SECUO,TRX99
      INTEGER*4 CTRLO,CTRL1
      INTEGER*4 DPDPS,DPQCS,EPDPS,EPQCS
      INTEGER*2 IDUM1,FACNO,POINT,POINTX,POINTY,IPARAM,NSORTD,EXITPT
      INTEGER*2 NDEPLC,NSECUR,NCUST,AGENCY,NIMMI,NPARKL,AIRLIN,XY
      INTEGER*2 ENTRPT,EXPCHK,BOARDT,WALKT,BUSTOP,TRFLTI,TRFLTO
      INTEGER*2 FLTNO,AC,DOM,COM,INT,GATE,PAX,BAG,DEFLIN
      INTEGER*2 IEPSCH,LINES,EPCURB,DEFBAG,STO,CAP,ADD,DELETE,SCALE
      INTEGER*2 AGENTS,SIZE,DIST,NO,TWOWAY,DOMDIR,COMDIR,INTDIR
      INTEGER*2 DPARK(4),CURBQ(4),TPAX(3)
      INTEGER*2 NPTCSW
      INTEGER*2 PH,PF,PB,PL,LR,LS
      INTEGER*2 NSNAP(20,2)/40*0/
      INTEGER*2 ITITLE(64,5)
      INTEGER*2 GPSTO(24)/24*0/,GPQUE(24)/24*0/,GPHALF(24)/24*0/
      INTEGER*2 ENTRCT(24)/24*0/,CRCON(24)/24*0/
       INTEGER*2 TSSOUT(25)/25*0/,TSQUE(25)/25*0/,TSHALF(25)/25*0/
      INTEGER*2 FLOW,ITIME,JENTCT,JCRCON,ISTRNO
      INTEGER*2 JTHLF(24)/24*0/,ISECFL(7)/7*0/,ITCKFL(15)/15*0/
      INTEGER*2 XENTCT,XCRCON,ISHLF,JSECFL,JTCKFL
       INTEGER*2 TTFLOW(16)/16*0/,TSFLOW(8)/8*0/
C
      DIMENSION BUFFER(21),ICARD(20),FACNO(4),FACTYP(20),IPARAM(3),XY(2)
      DIMENSION NFACSM(20,2),IDUM1(24),INDEXF(20),NSORTD(2),LINES(8)
      DIMENSION IEPSCH(10,10),NAMER8(20),CAP(15),XFRFLT(101),FACQSX(20)
       DIMENSION AGENTS(4),SIZE(4),DUM8(6),FROMTO(2),ITEMPA(24)
      DIMENSION ITEMPB(20)
      DIMENSION SERVRS(30)
C
      DATA NFACSM,DEFLIN,DEFBAG,BOARDT,ZAP/43*0,Z0000000000000000/
      DATA TRFLTI,TRFLTO,NO/2*0,'NO'/
       DATA ICHNG1/0/,ITEMPA/24*0/
      DATA PH,PF,PL,PB,LR,LS/'PH','PF','PL','PB','LR','LS',
      DATA IARRV,IDEPT,IOVER,IGRTR/'ARRV','DEPT','OVER','GRTR'/
      DATA IPARM,IBUS,ISTOR,ITRANS/'PARM','BUS/','STOR','TRAN'/
      DATA BLANK,JOBTST,IARLIN,IPRETI/'    ','JOBT','AIRL','%PRE'/
      DATA IRUNT,ICHAN,ITISER/'RUNT','CHAN','TIME'/
      DATA FACTYP/'GATE','CHEC','SECU','BAGC','CUST','ENTR','EXIT',
     *           'ENPL','XFER','PARK','RENT','DEPL','IMMI','TICK',
```

```
     *             'CONC',5*'    '/
      DATA NAMEFL,NAMEGE,NAMEGT,NAMEND/' &FL',' &GE',' &GT','&END'/
      DATA NAMEST,NAMEAL,NAMETI,NAMEOV/' &ST',' &AL',' &TI',' &OV'/
      DATA NAMEPA,NAMEBU,NAMES,NAMETR/' &PA',' &BU',' &S ',' &TR'/
      DATA NAMECH,NAMETS/' &CH',' &TS'/
      DATA BLANK1,MASK1,ASTRSK/Z40000000,ZFF000000,Z5C000000/
      DATA BLANK2,MASK2/Z40400000,Z0000FFFF/
      DATA IEPSCH/ 1,2,3,4,5,6,7,8,9,10,      2,1,3,4,5,6,7,8,9,10,
     *             3,2,4,1,5,6,7,8,9,10,      4,3,5,2,6,1,7,8,9,10,
     *             5,4,6,3,7,2,8,1,9,10,      6,5,7,4,8,3,9,2,10,1,
     *             7,6,8,5,9,4,10,3,2,1,      8,7,9,6,10,5,4,3,2,1,
     *             9,8,10,7,6,5,4,3,2,1,      10,9,8,7,6,5,4,3,2,1/
      DATA LINSNP,LINSNX,NTLINS/2*50,0/
      DATA NAMER8/'GATE    ','CHECKIN ','SECURITY','BAGCLAIM',
     *            'CUSTOMS ','ENTRANCE','EXIT    ','ENPLCURB',
     *            'TRANSFER','PARKING ','RENTACAR','DEPLCURB',
     *            'IMMIGRAT','TICKETS&','CONCESSI',5*'        '/
C
C
C
C    THIS EQUIVALENCE PROVIDES A CONVENIENT WAY TO ZERO OUT THE AREA OF
C    MAIN MEMORY CONTAINING INPUT VALUES BEFORE READING EACH CARD.
      EQUIVALENCE (DUM8(1),IDUM1(1),FACNO(1),BAG,LINES(1),STO,ADD),
     *            (IDUM1(2),CAP(1)),(IDUM1(3),FLTNO),
     *            (IDUM1(4),GATE),
     *            (IDUM1(5),TIME,AGENTS(1),SIZE(1)),
     *            (IDUM1(7),AC,DIST),       (IDUM1(8),DOM),
     *            (IDUM1(9),INT,EXITPT),    (IDUM1(10),COM,ENTRPT),
     *            (IDUM1(11),IPARAM(1),EPCURB,NDEPLC,NSECUR,NCUST,
     *                          AGENCY,DELETE,AIRLIN,DOMDIR),
     *            (IDUM1(12),PAX,EXPCHK,NIMMI,NPARKL,COMDIR),
     *            (IDUM1(13),BUSTOP,INTDIR,TPAX(1)),
     *            (IDUM1(14),POINT),
     *            (IDUM1(15),POINTX,XY(1)), (IDUM1(16),POINTY),
     *            (IDUM1(17),DPARK(1)),(IDUM1(21),CURBQ(1))
C
C    THIS EQUIVALENCE OVERLAYS COL. 1 OF "NFACSM" (THE NUMBERS OF EACH
C    FACILITY TYPE) WITH SCALARS WITH MORE INTELLIGABLE NAMES.
C    EXAMPLE: NFACSM(3,1) AND "NOSECU" BOTH REFER TO THE NUMBER OF
C             AIRPORT SECURITY FACILITIES.
      EQUIVALENCE (NFACSM(1,1),NOGATE),     (NFACSM(2,1),NOCHEC),
     *            (NFACSM(3,1),NOSECU),     (NFACSM(4,1),NOBAGC),
     *            (NFACSM(5,1),NOCUST),     (NFACSM(6,1),NOENTR),
     *            (NFACSM(7,1),NOEXIT),     (NFACSM(8,1),NOENPL),
     *            (NFACSM(9,1),NOTRAN),     (NFACSM(10,1),NOPARK),
     *            (NFACSM(11,1),NORENT),    (NFACSM(12,1),NODELP),
     *            (NFACSM(13,1),NOIMMI),    (NFACSM(14,1),NOTICK),
     *            (NFACSM(15,1),NOCONC),    (NFACSM(1,2),INDEXF(1))
C
C    THIS EQUIVALENCE OVERLAYS FACQSX WITH THE BASE VALUES OF THE
C    QUEUES, STORAGES, ETC., ASSIGNED EACH FACILITY TYPE BY THE GPSS
C    COMPILER.
      EQUIVALENCE (FACQSX(1),GAQSL),        (FACQSX(2),CHKQS),
     *            (FACQSX(3),SECQS),        (FACQSX(5),CUSQS),
     *            (FACQSX(8),EPCBS),        (FACQSX(10),PARQS),
     *            (FACQSX(11),RCRQS),       (FACQSX(12),DPCBS),
     *            (FACQSX(13),IMMQS),       (FACQSX(14),TICQS)
C
C    THIS EQUIVALENCE ENABLES THE BUILT-IN SORT ROUTINE TO SORT MH9 BY
C    FACILITY NUMBER BY FACILITY TYPE IN A SINGLE PASS.
      EQUIVALENCE (MSORT,NSORTD(1))
```

```
C
C     THIS EQUIVALENCE OVERLAYS THE ARRAY "FROMTO" WITH
C     "FROM" AND "TO".   SEE OVERRIDE CARD.
       EQUIVALENCE (FROMTO(1),FROM),          (FROMTO(2),TO)
C
C
C
C
C     THIS NAMELIST FOR "AIRLINE" CARDS.
       NAMELIST/AL/LINES,
     *              EPCURB,
     *              EXPCHK,
     *              BUSTOP
C
C     THIS NAMELIST FOR "BUS/LIMO" CARD.
       NAMELIST/BU/ARVBUS,
     *              DEPBUS
C
C     THIS NAMELIST FOR "ARRV" AND "DEPT" (FLIGHT) CARDS.
C     REQUIRED:  TIME, GATE, PAX.
C                FOR DEPARTING FLIGHTS; ARLIN OR DEFLIN.
C                FOR ARRIVING FLIGHTS; BAG OR DEFBAG.
C               SPECIFY MIDNIGHT AS 2400.
C     DEFAULTS: DOM=1,AIRLIN=DEFLIN,TPAX=0
C               BAG (CLAIM AREA) FOR ARRIVING FLIGHTS ONLY.
       NAMELIST/FL/FLTNO,
     *              AIRLIN,
     *              TIME,
     *              AC,
     *              DOM,INT,COM,
     *              GATE,
     *              PAX,
     *              TPAX,
     *              BAG
C
C     THIS NAMELIST FOR THE FOLLOWING FACILITY LOCATION CARDS:
C          "GATE"               "CHECKIN"             "SECURITY"
C          "BAGCLAIM"           "CUSTOMS"             "ENTRANCE"
C          "EXIT"               "ENPLCURB"            "XFER"
C          "PARKING"            "RENTACAR"            "DEPLCURB"
C          "IMMIGRATION"        "TICKETS&CHECKIN"
       NAMELIST/GE/FACNO,
     *              AGENTS,SIZE,TWOWAY,DPARK,CURBQ,
     *              IPARAM,
     *              NDEPLC,NSECUR,NCUST,AGENCY,AIRLIN,
     *              NIMMI,NPARKL,
     *              POINT,
     *              XY,POINTX,POINTY,
     *              EXITPT,
     *              ENTRPT
C
C     THIS NAMELIST FOR "GRTRANSP" (GROUND TRANSPORTATION) CARD.
C     REAL VARIABLES: PVTCAR, SELF, CRENT, BUS, TAXI, CURB, PARK
       NAMELIST/GT/DOM,COM,INT,
     *               PVTCAR,
     *               CRENT,
     *               BUS,
     *              TAXI
C
C     THIS NAMELIST FOR WALKING TIME "OVERRIDE" CARDS.
       NAMELIST/OV/FROMTO,FROM,TO,
```

```
     *           TIME,DIST
C
C    THIS NAMELIST FOR "PARM" (PARAMETER) CARDS.
C    REAL VARIABLES: GREET, WWGATE, GRGATE, CIRCPK
C    DEFAULTS: BOARDT= 15 MIN., ERRORS = 50.
      NAMELIST/PA/ERRORS,
     *           BOARDT,
     *           LEAVEL,
     *           LEAVEC,
     *           LEAVEV,
     *           GREET,
     *           WWGATE,
     *           GRGATE,
     *           CURBCK,
     *            CIRCPK,
     *           CRBGT,
     *           PRKCRB
C
C    THIS NAMELIST FOR "STORAGE" CARDS.
      NAMELIST/S/STO,
     *          CAP
C
C    THIS NAMELIST FOR "INITIAL" CARD.
C    REAL VARIABLES: DSTFAC AND WALKSP ARE REAL VARIABLES.
C    DEFAULTS: SCALE=1, DSTFAC=1.1, WALKSP=1.0(METERS/SEC).
      NAMELIST/ST/START,
     *           FINISH,
     *           DEFBAG,
     *           DEFLIN,
     *           DSTFAC,
     *           SCALE,
     *           WALKSP
C
C    THIS NAMELIST FOR "%PRETICKETED" CARD.
      NAMELIST/TI/DOM,
     *           COM,
     *           INT,
     *           DOMDIR,
     *           COMDIR,
     *           INTDIR
C
C    THIS NAMELIST FOR "TRANSFER" (FLIGHT) CARD.
C    DEFAULTS: ADD=7200(120 MIN),DELETE=1800(30 MIN).
      NAMELIST/TR/ADD,
     *           DELETE
C
C    THIS NAMELIST FOR "CHANGE" CARD.
      NAMELIST/CH/TIME,
     *           SERVRS
C
C
C     THIS NAMELIST IS FOR ARRAYS SPECIFYING
C     STORAGE QUEUE AND HALF-WORD NUMBERS FOR
C     TIME SERIES READ-IN ON 'TIME-SERIES' RECORD.
C
      NAMELIST/TS/GPSTO,
     *           GPQUE,
     *           GPHALF
C
      MH1(IR,IC)=MH01B+ICNH01*IR+IC
      MH2(IR,IC)=MH02B+ICNH02*IR+IC
```

```
      MH3(IR,IC)=MH03B+ICNH03*IR+IC
      MH4(IR,IC)=MH04B+ICNH04*IR+IC
      MH5(IR)=MH05B+IR+1
      MH6(IR,IC)=MH06B+ICNH06*IR+IC
      MH7(IR,IC)=MH07B+ICNH07*IR+IC
      MH8(IR,IC)=MH08B+ICNH08*IR+IC
      MH9(IR,IC)=MH09B+ICNH09*IR+IC
      MH11(IR)=MH11B+IR+1
      MH12(IR)=MH12B+IR+1
      MH13(IR)=MH13B+IR+1
      ML2(IR,IC)=ML02B+ICNLC2*IR+IC
C
C
      RETURN
C
C
C
C
      ENTRY FORTM(IVALUE)
C
C
      IBRNCH=IVALUE(1)
      GOTO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
     *     21,22,23,24,25),IBRNCH
C
C
C
C
    1         C    O    N    T    I    N    U    E
C
C    PERFORM LINKAGES.
C    COMPUTE MATRIX BASE ADDRESSES.
C    READ SIMULATION START AND END TIMES.
C    RETURN LENGTH OF RUN IN XHSENDXH.
C
C
C...I N P U T    S E C T I O N
C
      WRITE(6,1005)
      JOBT=0
      LINECT=1
      MAXPT=0
      NCARD=0
      NERCNT=0
      NPTOSW=0
      NGEO=0
      NOFXFR=0
      NRCRSW=0
      NROW=0
      TRFLTI=0
      TRFLTO=0
      ITIME=1
      BUFFER(21)=NAMEND
C
C
C    D E F A U L T    V A L U E S    H E R E
C
      BOARDT=15
      DSTFAC=1.1
      ERRORS=50
      SCALE=1
```

```
      WALKSP=1.0
      ADD=120
      DELETE=30
      ARVBUS = 0
      DEPBUS = 0
      START = 0
      FINISH = 0
      FROM = 0
      TO = 0
      WWGATE = 0.0
      GRGATE = 0.0
      GREET=0.0
      CURBCK=0.0
      CIRCPK=0.0
         CRBGT=0.0
         PRKCRB=0.0
C  LATEST TIME LEAVE LOBBY CONCESSION
      LEAVEL = 15
C  LATEST TIME LEAVE CONCOURSE CONCESSION
      LEAVEC = 10
C  SPREAD OF UNIFORM DISTRIBUTION BEFORE ABOVE TIMES
      LEAVEV = 10
C
  117 READ(5,1002)ICARD
      NCARD=NCARD+1
      IF(ICARD(1).NE.JOBTST)GOTO 111
      JOBT=1
      WRITE(6,1004)NCARD,ICARD
      READ(5,1002)ICARD
      NCARD=NCARD+1
      WRITE(9,1002)ICARD
  111 WRITE(6,1004)NCARD,ICARD
      TYPTST=IAND(ICARD(1),MASK1)
      IF(TYPTST.EQ.ASTRSK)GOTO 117
      ICARD(1)=NAMEST
      ICARD(2)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,ST)
      IF(JOBT.EQ.1)GOTO 108
      CALL MNLINK(1,ICNH01,ICNH02,ICNH03,ICNH04,ICNH06,ICNH07,ICNH08,
     1    ICNH09,
     2    ICNL02,
     3    ENDXF,TRVXH,BDTXH,ABUXH,DBUXH,XFRXH,XFAXH,XFDXH,SCLXH,CLKXH,
     4    CUSQS,RCRQS,DPCBS,EPCBS,CHKQS,SECQS,GAQSL,PARQS,IMMQS,TICQS,
     5    RCARO,BAGCO,DPLCO,CHEK2,CHEK3,CGTRO,ERROR,SECUO,CTRLO,CTRL1,
     6    TRX99,CONXH,CHGXF,NSCXH,SLCXH,DPDPS,DPQCS,WWGXH,GRGXL,EPDPS,
     7    EPQCS, GRT00,GRTXL,CPKXH,CRBXH,CGTXL,PCBXL,
     8    JOBLS)
      CALL CLINK2
      ISAVEH(XFAXH)=ADD*60
      ISAVEH(XFDXH)=DELETE*60
      ISAVEH(SCLXH)=SCALE
C
C   DEFAULTS FOR ADDING, DELETING TRANSFER FLT FROM XFRFLT: 2HRS., 30 MI
C
      MH01B=MHBASE(IMAXH,1,ICNH01)
      MH02B=MHBASE(IMAXH,2,ICNH02)
      MH03B=MHBASE(IMAXH,3,ICNH03)
      MH04B=MHBASE(IMAXH,4,ICNH04)
```

```
      MH05B=MHBASE(IMAXH,5,1)
      MH06B=MHBASE(IMAXH,6,ICNH06)
      MH07B=MHBASE(IMAXH,7,ICNH07)
      MH08B=MHBASE(IMAXH,8,ICNH08)
      MH09B=MHBASE(IMAXH,9,ICNH09)
      MH11B=MHBASE(IMAXH,11,1)
      MH12B=MHBASE(IMAXH,12,1)
      MH13B=MHBASE(IMAXH,13,1)
      ML02B=MLBASE(IMAXL,2,ICNL02)
      GOTO 109
C
  108 CALL MNLINK(1,ICNH01,ICNH04,ICNL02,CLKXH)
      CALL CLINK2
      MH01B=MHBASE(IMAXH,1,ICNH01)
      MH04B=MHBASE(IMAXH,4,ICNH04)
      ML02B=MLBASE(IMAXL,2,ICNL02)
C
  109 ISAVEH(CLKXH)=START
      NSTHR=START/100
      NSTMIN=MOD(START,100)
      NENDHR=FINISH/100
      NENDMN=MOD(FINISH,100)
      IF(NENDMN.GE.NSTMIN)GOTO 100
      NENDHR=NENDHR-1
      NENDMN=NENDMN+60
  100 IF(JOBT.NE.1)ISAVEF(ENDXF)=60*(60*(NENDHR-NSTHR)+NENDMN-NSTMIN)-1
      GOTO 101
C
  101 DO 112 I=1,6
  112 DUMB(I) = ZAP
      TWOWAY=BLANK
      READ(5,1002,END=99)ICARD
      NCARD=NCARD+1
      LINECT=LINECT+1
      IF(LINECT.LT.51)GOTO 107
      LINECT=1
      WRITE(6,1005)
  107 WRITE(6,1004)NCARD,ICARD
      IF(JOBT.EQ.1)WRITE(9,1002)ICARD
      TYPTST=IAND(ICARD(1),MASK1)
      IF(TYPTST.EQ.ASTRSK)GOTO 101
      IF(ICARD(1).EQ.IARRV.OR.ICARD(1).EQ.IDEPT)GOTO 106
      IF(ICARD(1).EQ.IGRTR)GOTO 180
      IF(ICARD(1).EQ.ITISEP) GO TO 120
      IF(ICARD(1).EQ.IARLIN)GOTO 160
      IF(ICARD(1).EQ.IPRETI)GOTO 188
      IF(ICARD(1).EQ.IOVER)GOTO 170
      IF(ICARD(1).EQ.IPARM)GOTO 173
      IF(ICARD(1).EQ.IBUS)GOTO 186
      IF(ICARD(1).EQ.ISTOR)GOTO 190
      IF(ICARD(1).EQ.ITRANS)GOTO 195
      IF(ICARD(1).EQ.IRUNT)GOTO 200
      IF(ICARD(1).EQ.ICHAN)GOTO 99
C
      DO 102 I=1,20
         IF(FACTYP(I).EQ.BLANK)GOTO 104
         IF(FACTYP(I).EQ.ICARD(1))GOTO 215
  102 C O N T I N U E
C
C   ERROR IN FLIGHT INPUT DATA.
C
```

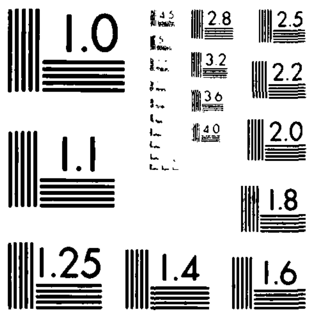
```
  199 WRITE(6,1000)
      CALL ASSIGN(1,1000,PH)
      NERRSW=1
      GOTO 101
C
C   ERROR IN GEOMETRY INPUT DATA.
C
  104 WRITE(6,1003)NCARD
      NERRSW=1
      CALL ASSIGN(1,1000,PH)
      GOTO 101
C
C...F L I G H T   S C H E D U L E   I N P U T
C
  106 CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      BUFFER(1)=NAMEFL
      CALL XCODE(BUFFER,84)
      READ(10,FL)
      NROW=NROW+1
      IF(GATE.EQ.0.OR.PAX.EQ.0.OR.TIME.EQ.0)GOTO 199
      IF(ICARD(1).EQ.IARRV)GOTO 113
      IF(DEFLIN.EQ.0.AND.AIRLIN.EQ.0)GOTO 199
      IMAXBH(MH1(NROW,1))=1
  113 IMAXBH(MH1(NROW,2))=FLTNO
      IF(AIRLIN.EQ.0)AIRLIN=DEFLIN
      IMAXBH(MH1(NROW,3))=AIRLIN
      IMAXBH(MH1(NROW,4))=TIME
      NFLTHR=TIME/100
      NFLTMN=MOD(TIME,100)
      IF(NFLTMN.GE.NSTMIN)GOTO 103
      NFLTHR=NFLTHR-1
      NFLTMN=NFLTMN+60
  103 IMAXBH(MH1(NROW,6))=60*(NFLTHR-NSTHR)+NFLTMN-NSTMIN
      IF(INT.NE.1)GOTO 105
      IMAXBH(MH1(NROW,7))=3
      GOTO 115
  105 IF(COM.NE.1)GOTO 110
      IMAXBH(MH1(NROW,7))=2
      GOTO 115
  110 IMAXBH(MH1(NROW,7))=1
  115 IMAXBH(MH1(NROW,8))=AC
      IMAXBH(MH1(NROW,9))=GATE
      IF(BAG.EQ.0.AND.ICARD(1).EQ.IARRV)BAG=DEFBAG
      IF(ICARD(1).EQ.IARRV.AND.BAG.EQ.0)GOTO 199
      IMAXBH(MH1(NROW,12))=BAG
      PAX = PAX-TPAX(1)-TPAX(2)-TPAX(3)
      IF(SCALE.EQ.1)GOTO 114
      IMAXBH(MH1(NROW,10))=PAX/SCALE+0.51
      IMAXBH(MH1(NROW,11))=TPAX(1)/SCALE+0.51
      IMAXBH(MH1(NROW,13))=TPAX(2)/SCALE+0.51
      IMAXBH(MH1(NROW,16))=TPAX(3)/SCALE+0.51
      GOTO 101
  114 IMAXBH(MH1(NROW,10))=PAX
      IMAXBH(MH1(NROW,11))=TPAX(1)
      IMAXBH(MH1(NROW,13))=TPAX(2)
      IMAXBH(MH1(NROW,16))=TPAX(3)
      GOTO 101
C
C...T I M E   S E R I E S   S P E C I F I C A T I O N S
C
```

```
C   TIME SERIES ENTITY SPECIFICATION PLACES NEMBERS
C   OF STORAGES, QUEUES AND HALF-WORDS IN GPSTO, GPQUE
C   AND GPHALF ARRAYS FOR TIME-SERIES READOUTS.
C
  120 ICARD(1)=NAMETS
      ICARD(2)=BLANK
      ICARD(3)=IAND(ICARD(3),MASK2)+BLANK2
      CALL XCODE(BUFFER,80)
      WRITE(10,1002) ICARD
      CALL XCODE(BUFFER,84)
      READ(10,TS)
      GO TO 101
C
C
C...A I R L I N E   D A T A   I N P U T
C
  160 IF(JOBT.EQ.1)GOTO 101
      ICARD(1)=NAMEAL
      ICARD(2)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,AL)
      DO 163 I=1,8
         J=LINES(I)
         IF(J.EQ.0)GOTO 101
         IMAXBH(MH2(J,1))=EPCURB
         IMAXBH(MH2(J,2))=EXPCHK*10
         IMAXBH(MH2(J,3))=BUSTOP
  163 C O N T I N U E
      GOTO 101
C
C...G R O U N D   T R A N S P O R T   I N P U T
C
  180 PVTCAR=0.0
      CRENT=0.0
      BUS=0.0
      TAXI=0.0
      ICARD(1)=NAMEGT
      ICARD(2)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,GT)
       PVTCAR=PVTCAR/100.
      CRENT=CRENT/100.
      BUS=BUS/100.
      TAXI=TAXI/100.
      IF(DOM.NE.1)GOTO 181
      I=1
      GOTO 183
  181 IF(COM.NE.1)GOTO 182
      I=2
      GOTO 183
  182 I=3
  183 IF(JOBT.EQ.1)GOTO 184
C   NORMAL MODE - DEPL PAX LOGIC
C   ML2(1-3,2-*) A CUM PROB DIST WITH PVT CAR ELIMINATED
      TEMPCT = 1.0 - PVTCAR
       FMAXBL(ML2(I,1)) = PVTCAR
C
```

```
      TEMP2 = CRENT/TEMPCT
      FMAXBL (ML2(I,2))=TEMP2
      TEMP2=TEMP2+BUS/TEMPCT
      FMAXBL (ML2(I,3))=TEMP2
      FMAXBL (ML2(I,4))=1.0
      GOTO 101
C   USED BY SATELITE PROGRAM WHEN CREATING ENPL PAX JOBTAPE
  184 FMAXBL(ML2(I,1))=PVTCAR
      TEMP2=PVTCAR+CRENT
      FMAXBL(ML2(I,2))=TEMP2
      TEMP2=TEMP2+BUS
      FMAXBL(ML2(I,3))=TEMP2
      FMAXBL(ML2(I,4))=1.0
      GOTO 101
C
C...% P R E T I C K E T E D   P A X   I N P U T
C
  188 ICARD(1)=NAMETI
      ICARD(2)=BLANK
      ICARD(3)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,TI)
      IMAXBH(MH4(1,1))=DOM*10
      IMAXBH(MH4(2,1))=COM*10
      IMAXBH(MH4(3,1))=INT*10
      IF(DOMDIR.GT.0.AND.DOM.GT.0) IMAXBH(MH4(1,2))=DOMDIR*10
      IF(COMDIR.GT.0.AND.COM.GT.0) IMAXBH(MH4(2,2))=COMDIR*10
      IF(INTDIR.GT.0.AND.INT.GT.0) IMAXBH(MH4(3,2))=INTDIR*10
      GOTO 101
C
C...W A L K I N G   T I M E / D I S T   O V E R R I D E   I N P U T
C
  170 IF(JOBT.EQ.1)GOTO 101
      ICARD(1)=NAMEOV
      ICARD(2)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,OV)
      IF(TIME.GT.0)GOTO 171
      TIME=DIST/WALKSP
  171 IMAXBH(MH6(FROM,TO))=TIME
      IMAXBH(MH6(TO,FROM))=TIME
      GOTO 101
C
C...P A R M   C A R D S   I N P U T
C
  173 IF(JOBT.EQ.1)GOTO 101
      ICARD(1)=NAMEPA
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,PA)
      GOTO 101
C
C...B U S   S C H E D U L E   I N P U T
C
  186 IF(JOBT.EQ.1)GOTO 101
      ICARD(1)=NAMEBU
```

AD-A117 603 TRANSPORTATION SYSTEMS CENTER CAMBRIDGE MA F/G 1/5
AIRPORT LANDSIDE. VOLUME V. APPENDIX B. ALSIM SUBROUTINES.(U)
JUN 82 L MCCABE, M GORSTEIN
UNCLASSIFIED DOT-TSC-FAA-82-4-5 FAA-EM-80-8-5 NL

3 OF 4
AD
A117603




MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

```
      ICARD(2)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,BU)
      ISAVEH(ABUXH)=60.*ARVBUS
      ISAVEH(DBUXH)=60.*DEPBUS
      GOTO 101
C
C...G P S S   S T O R A G E   C A P A C I T Y   I N P U T
C
  190 IF(JOBT.EQ.1)GOTO 101
      ICARD(1)=NAMES
      ICARD(2)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,S)
      DO 191 I=1,15
         IF(CAP(I).EQ.0)GOTO 101
         ISTO(11*(STO+I-2)+2)=CAP(I)
  191 C O N T I N U E
      GOTO 101
C
C...T R A N S F E R   F L I G H T   O V E R R I D E S   I N P U T
C
  195 IF(JOBT.EQ.1)GOTO 101
      ICARD(1)=NAMETR
      ICARD(2)=BLANK
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,TR)
      IF(ADD.GT.0)ISAVEH(XFAXH)=ADD*60
      IF(DELETE.GT.0)ISAVEH(XFDXH)=DELETE*60
      GOTO 101
C
C...R U N T I T L E   C A R D   I N P U T
C
  200 IF(JOBT.EQ.1)GOTO 101
      IF(NTLINS.LT.5)GOTO 201
      WRITE(6,1080)
      GOTO 101
  201 NTLINS=NTLINS+1
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,80)
      READ(10,1081)(ITITLE(I,NTLINS),I=1,64)
      GOTO 101
C
C...G E O M E T R Y   I N P U T
C
  215 IF(JOBT.EQ.1)GOTO 101
C   SET J = ENTITY RANGE FOR FAC. TYPE -1.   ISTO(N-1) ACCOUNTS FOR 2ND
      J=FACOSX(I)-2
      NOFAC=I
      ICARD(1)=NAMEGE
      TYPTST=IAND(ICARD(2),MASK1)
      IF(TYPTST.NE.BLANK1)ICARD(2)=BLANK
C   CHECK FOR "IMMIGRATION" & "TICKETS&CHECKIN"
      IF(NOFAC.EQ.13.OR.NOFAC.EQ.14)ICARD(3)=BLANK
```

```
      IF(NOFAC.EQ.14)ICARD(4)=BLANK
      IF (NOFAC.EQ.15) ICARD(3) = IAND(ICARD(3),MASK2)+BLANK2
      CALL XCODE(BUFFER,80)
      WRITE(10,1002)ICARD
      CALL XCODE(BUFFER,84)
      READ(10,GE)
      IF(NERRSW.EQ.1)GOTO 101
C   ARGUMENTS TO FUNCTION MH3 MUST BE I*4.
      I=POINT
      IF(POINTX.NE.0)IMAXBH(MH3(I,1))=POINTX
      IF(POINTY.NE.0)IMAXBH(MH3(I,2))=POINTY
      IF(EXITPT.GT.0)IMAXBH(MH3(I,3))=EXITPT
      IF(ENTRPT.GT.0)IMAXBH(MH3(I,4))=ENTRPT
C
      DO 225 I=1,4
         IF(FACNO(I).EQ.0)GOTO 227
         NGEO=NGEO+1
         NFACSM(NOFAC,1)=NFACSM(NOFAC,1)+1
         IMAXBH(MH9(NGEO,1))=NOFAC
         IMAXBH(MH9(NGEO,2))=FACNO(I)
         IMAXBH(MH9(NGEO,3))=POINT
         IF(POINT.GT.MAXPT)MAXPT=POINT
         IF(SIZE(I).EQ.0)GOTO 220
         K=11*(J+FACNO(I))+2
         ISTO(K)=SIZE(I)
 220     IF (NOFAC.NE.8) GO TO 221
C  ENPLANING CURB SPECIAL TREATMENT
      ISTO(K) =  SIZE(I)/SCALE+0.5
         IF (ISTO(K).EQ.0) ISTO(K) = 1
         K = 11*(EPDPS+FACNO(I)-2)+2
      ISTO(K) =  DPARK(I)/SCALE+0.5
         IF (ISTO(K).EQ.0) ISTO(K) = 1
         K = 11*(EPOCS+FACNO(I)-2)+2
      ISTO(K) =  CURBQ(I)/SCALE+0.5
         IF (ISTO(K).EQ.0) ISTO(K) = 1
         GO TO 222
 221     IF (NOFAC.NE.12) GO TO 222
C  DEPLANING CURB SPECIAL TREATMENT
      ISTO(K) =  SIZE(I)/SCALE+0.5
         IF (ISTO(K).EQ.0) ISTO(K) = 1
         K = 11*(DPDPS+FACNO(I)-2)+2
      ISTO(K) =  DPARK(I)/SCALE+0.5
         IF (ISTO(K).EQ.0) ISTO(K) = 1
         K = 11*(DPOCS+FACNO(I)-2)+2
      ISTO(K) =  CURBQ(I)/SCALE+0.5
         IF (ISTO(K).EQ.0) ISTO(K) = 1
 222     DO 223 M=1,3
            IF(IPARAM(M).EQ.0)GOTO 225
            L=M+3
            IMAXBH(MH9(NGEO,L))=IPARAM(M)
 223     C O N T I N U E
 225  C O N T I N U E
C   CHECK FOR SIDE BY SIDE ENTRANCE/EXIT
 227  IF(NOFAC.NE.6.AND.NOFAC.NE.7.OR.TWOWAY.EQ.NO)GOTO 101
      TWOWAY=NO
      I=13-NOFAC
      GOTO 215
C
C
  99  IF(NERRSW.EQ.1)GOTO 9 9 9 9 9
C
```

```
C   SORT FLIGHT SCHEDULE ON COL 6.
C
      K=NROW-1
      DO 230 J=1,K
         NTEST=IMAXBH(MH1(J,6))
         IF(NTEST.EQ.0)GOTO 230
         DO 229 I=J,NROW
            NTIME=IMAXBH(MH1(I,6))
            IF(NTIME.EQ.0.OR.NTIME.GE.NTEST)GOTO 229
            NTEST=NTIME
            DO 228 L=1,ICNH01
               ITEMP1=IMAXBH(MH1(J,L))
               IMAXBH(MH1(J,L))=IMAXBH(MH1(I,L))
               IMAXBH(MH1(I,L))=ITEMP1
  228       C O N T I N U E
  229    C O N T I N U E
  230 C O N T I N U E
C
C   MARK COL 1 FOR PAST LAST FLIGHT.
C
      IMAXBH(MH1(NROW+1,1))=-1
      IF(JOBT.EQ.1)GOTO 299
C
C   SORT BY FACILITY NUMBER, THEN FACILITY TYPE.
C
      NSWTC1=0
  251 K=NGEO-1
      DO 260 J=1,K
         NTEST=2147483647
         DO 255 I=J,NGEO
            NSORTD(1)=IMAXBH(MH9(I,1))
            NSORTD(2)=IMAXBH(MH9(I,2))
            IF(NSORT.GE.NTEST)GOTO 255
            NTEST=NSORT
            MINROW=I
  255    C O N T I N U E
         IF(MINROW.EQ.J)GOTO 260
         DO 254 M=1,6
            ITEMP1=MH9(MINROW,M)
            ITEMP2=MH9(J,M)
            ITEMP3=IMAXBH(ITEMP1)
            IMAXBH(ITEMP1)=IMAXBH(ITEMP2)
            IMAXBH(ITEMP2)=ITEMP3
  254    C O N T I N U E
  260 C O N T I N U E
  265 IF(NSWTC1.EQ.1)GOTO 290
      ITEMP1=0
      DO 280 I=1,20
         IF(FACTYP(I).EQ.BLANK)GOTO 295
         IF(NFACSM(I,1).EQ.0)GOTO 280
         ITEMP1=ITEMP1+NFACSM(I,1)
         IF(IMAXBH(MH9(ITEMP1,2)).EQ.NFACSM(I,1))GOTO 280
         NSWTC1=1
         ITEMP2=ITEMP1-NFACSM(I,1)+1
         ITEMP3=0
         DO 270 J=ITEMP2,ITEMP1
  268       ITEMP3=ITEMP3+1
C     CHECK FOR DOUBLY DEFINED FACILITY
         IF(IMAXBH(MH9(J,2)).LT.ITEMP3)GOTO 289
            IF(IMAXBH(MH9(J,2)).EQ.ITEMP3)GOTO 270
C     BUILD DUMMY FACILITIES
```

```
            NGEO=NGEO+1
            IMAXBH(MH9(NGEO,1))=I
            IMAXBH(MH9(NGEO,2))=ITEMP3
            NFACSM(I,1)=NFACSM(I,1)+1
            GOTO 268
  269     K=IMAXBH(MH9(J,2))
          WRITE(6,1031)NAMER8,I
          CALL ASSIGN(1,1000,PH)
          GOTO 9 9 9 9 9
  270     C O N T I N U E
  280 C O N T I N U E
  295 IF(NSWTC1.EQ.1)GOTO 251
C
C   INDEXF(*) (NFACSM(*,2) + NO OF FAC IN THAT TYPE POINTS TO FAC IN FAC
C
  290 NFACSM(1,2)=0
      IMAXBH(MH8(1,1))=NFACSM(1,1)
      DO 296 I=2,20
         J=I-1
         NFACSM(I,2)=NFACSM(J,1)+NFACSM(J,2)
         IMAXBH(MH8(I,1))=NFACSM(I,1)
         IMAXBH(MH8(I,2))=NFACSM(I,2)
  296 C O N T I N U E
C
C   POINT-TO-POINT WALKING TIME CALCULATION
C
      WRITE(6,1024)
      DO 293 I=1,MAXPT
         X1=IMAXBH(MH3(I,1))
         Y1=IMAXBH(MH3(I,2))
         IF(X1.EQ.0.0.AND.Y1.EQ.0.0)WRITE(6,1025)I
C       TEST FOR POINTX, POINTY BOTH 0 ---> PROBABLY NOT DEFINED.
         DO 292 J=I,MAXPT
            IF(I.EQ.J)GOTO 292
            K=MH6(I,J)
            IF(IMAXBH(K).GT.0)GOTO 292
            X2=IMAXBH(MH3(J,1))
            Y2=IMAXBH(MH3(J,2))
            ITEMP1=SQRT((X1-X2)**2+(Y1-Y2)**2)*DSTFAC/WALKSP
            IMAXBH(K)=ITEMP1
            IMAXBH(MH6(J,I))=ITEMP1
  292    C O N T I N U E
  293 C O N T I N U E
C
C   DETERMINE CLOSEST EXIT & ENTRANCE TO EACH POINT.
C
      I=INDEXF(7)
      J=I+NOEXIT
      NCOL=3
  294 I=I+1
      DO 298 K=1,MAXPT
         IF(IMAXBH(MH3(K,NCOL)).NE.0)GOTO 298
         ITEMP1=9999
         DO 297 L=I,J
            ITEMP4=IMAXBH(MH9(L,3))
            ITEMP2=IMAXBH(MH6(ITEMP4,K))
            IF(ITEMP2.GE.ITEMP1)GOTO 297
            ITEMP1=ITEMP2
            ITEMP3=ITEMP4
  297    C O N T I N U E
         IMAXBH(MH3(K,NCOL))=ITEMP3
```

```
  298 C O N T I N U E
      IF(NCOL.EQ.4)GOTO 291
      NCOL=4
      I=INDEXF(6)
      J=I+NOENTR
      GOTO 294
C
C   CHECK FOR UNDEFINED FACILITIES - NOT NECESSARILY AN ERROR CONDITION.
C
  291 NSWTC1=0
      DO 285 I=1,20
         IF(FACTYP(I).EQ.BLANK)GOTO 286
         IF(NFACSM(I,1).GT.0)GOTO 285
         NSWTC1=1
  285 C O N T I N U E
  286 IF(NSWTC1.EQ.0)GOTO 289
      WRITE(6,1020)
      DO 287 I=1,20
         IF(FACTYP(I).EQ.BLANK)GOTO 288
         IF(NFACSM(I,1).GT.0)GOTO 287
         IF(I.NE.14)GOTO 283
         WRITE(6,1026)
         GOTO 287
  283    IF(I.NE.13)GOTO 284
         WRITE(6,1030)
         GOTO 287
  284    WRITE(6,1021)NAMERB(I)
  287 C O N T I N U E
  288 WRITE(6,1022)
  289 CONTINUE
C
C  PARAMETER INPUT VALUES PLACED IN GPSS PROGRAM
C
      ISAVEH(BDTXH)=60*BOARDT
      LEAVEL = 60*LEAVEL
      LEAVEC = 60*LEAVEC
      LEAVEV = 60*LEAVEV
      ISAVEH(WWGXH) = 10.*WWGATE+0.5
      FSAVEL(GWGXL) = GRGATE/100.
      FSAVEL(CGTXL)=CRBGT/100.
      FSAVEL(PCBXL)=PRKCRB/100.
      FSAVEL(GRTXL) = GRFET/100.
      ISAVEH(CPKXH) = 10.*CIRCPK+0.5
      ISAVEH(CRBXH) = 10.*CURBCK+0.5
C
C
C   MESSAGE - E N D   O F   I N P U T   D A T A
C
  299 WRITE(6,1006)
      GOTO 9 9 9 9
C
C
C
C
    2       C     O     N     T     I     N     U     E
C
C...B A G G A G E   U N L O A D   R O U T I N E
C
C   SCANS MH7, BUILT BY SUCCESSIVE CALLS TO "BAGS" BY PAX XACS.   LOADS
C   SUCCESSIVE PB'S WITH NUMBER TO BE COMPARED WITH MAX RANDOM
C   NUMBER OF PASSENGERS ON FLIGHT USER CHAIN.   ROUTINE WILL LOOP
```

```
C    THROUGH LOGIC CONTAINING A TIME DELAY, UNLINKING ALL PASSENGERS
C    WITH MAX RANDOM NUMBERS LE SUCCESSIVE PB* VALUES.
C
C    MH7(2,1)=MH7(2,1)+MH7(1,1)
C    MH7(3,1)=MH7(3,1)+MH7(2,1)
C    ETC.
C
      MAXBAG=IVALUE(2)
      NTEST=MAXBAG
      NOPB=40
      NENDCK=0
      ITEMP1=MH07B+1
      DO 305 I=1,63
         ITEMP1=ITEMP1+1
         ITEMP2=ITEMP1+1
         NOBAGS=IMAXBH(ITEMP1)
         IMAXBH(ITEMP1)=0
         IF(NENDCK.EQ.0.)NENDCK=NOBAGS
         IMAXBH(ITEMP2)=IMAXBH(ITEMP2)+NOBAGS
         IF(IMAXBH(ITEMP2).LT.NTEST)GOTO 305
         CALL ASSIGN( NOPB,I+1,PB )
         NENDCK=0
         IF (NOPB.EQ.1) GO TO 306
         NOPB=NOPB-1
         NTEST=NTEST+MAXBAG
  305 C O N T I N U E
      IMAXBH(ITEMP2)=0
      IF(NENDCK.EQ.0)GOTO 9 9 9 9 9
  306 CALL ASSIGN(NOPB,64,PB)
      GOTO 9 9 9 9 9
C
C
    3        C     O     N     T     I     N     U     E
C
C...B A G C L A I M
C
C                   IVALUE(2) = CURRENT LOCATION
C                   IVALUE(3) = PH1 (MH1 ROW NO)
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
      J=INDEXF(4)+IMAXBH(MH1(IV3,12))
      NPTTO=IMAXBH(MH9(J,3))
      ASSIGN 309 TO NEXT
      GOTO 950
  309 CALL ASSIGN( 2,NPTTO,PH, 11,4,PB, 7,J,PH )
      GOTO 9 9 9 9 9
C
    4        C     O     N     T     I     N     U     E
C
C
C...C U S T O M S
C
C                   IVALUE(2) = CURRENT LOCATION
C                   IVALUE(3) = MH9 ROW NO OF APPROPRIATE IMMIGRATION FAC
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
C    CUSTOMS AREA ASSOCIATED WITH IMMIGRATION AREA PAX AT
      L=IMAXBH(MH9(IV3,4))
      J=INDEXF(5)+L
```

```
      NPTTO=IMAXBH(MH9(J,3))
      ASSIGN 313 TO NEXT
      GOTO 950
C   DETERMINE CUSTOMS QUEUE AND STORAGE NUMBER
  313 M=CUSQS+L-1
      CALL ASSIGN( 2,NPTTO,PH, 5,M,PH, 7,J,PH, 11,5,PB )
C
      GOTO 9 9 9 9 9
C
C
    5       C     O     N     T     I     N     U     E
C
C...G R O U N D   T R A N S P O R T   M O D E
C
C                 IVALUE(2) = PAX BEING MET  (DEPL PAX; DECR TO 0 BY ROU
C                           = RANDOM NO FOR TICKETED/NOT SELECTION FOR J
C                 IVALUE(3) = RANDOM NO FOR MODE SELECTION
C                 IVALUE(4) = FLT TYPE (1,2,3 = DOM,COM,INT)
C
      IV2=IVALUE(2)
      IV4=IVALUE(4)
      IF(JOBT.EQ.1)GOTO 702
      K = 2
      L = 0
      GO TO 701
C   PAX NOT BEING MET; RANDOM MODE SELECTION
  702 K=1
      L=0
C   DECISION ON TICKETED/NOT TICKETED
      IF(IMAXBH(MH4(IV4,1)).LT.IV2)L=1
  701 TEMPCT=(IVALUE(3)+1.)/1000.
      DO 705 J=K,10
         IF(TEMPCT.GT.FMAXBL(ML2(IV4,J)))GOTO 705
C   ADD (+1) TO J BECAUSE PVT.CAR PASS. GROUPS USE PB6=1 AND PB6=2
      J=J+1
         CALL ASSIGN( 6,J,PB, 9,L,PB )
         GOTO 9 9 9 9 9
  705 C O N T I N U E
      NERCNT=NERCNT+1
      IF(NERCNT.EQ.ERRORS)GOTO 999
      WRITE(6,1007)
      CALL ASSIGN( 6,4,PB, 9,L,PB )
      GOTO 9 9 9 9 9
C
C
    6       C     O     N     T     I     N     U     E
C
C...R E N T A C A R
C
C                   IVALUE(2) = CURRENT LOCATION - PH2
C                   IVALUE(3) = CAR RENTAL AGENCY CODE - PB10
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
C   ITEMP1 = DIST TO CLOSEST COUNTER OF AGENCY,
C   MINPTO = CLOSEST COUNTER'S POINT NUMBER.
      I=INDEXF(11)
      J=I+NORENT
      I=I+1
C   SCAN AGENCY COUNTERS TO FIND NEAREST ONE OF CORRECT AGENCY
      ITEMP1=99999
```

```
      MINPTO=0
      LTEMP=0
      DO 320 N=I,J
         LTEMP=LTEMP+1
C        BRANCH IF DIFFERENT AGENCY
         IF(IMAXBH(MH9(N,4)).NE.IV3)GOTO 320
         NPTTO=IMAXBH(MH9(N,3))
         ITEMP2=IMAXBH(MH6(NPTFM,NPTTO))
C        BRANCH IF NOT CLOSEST COUNTER.
         IF(ITEMP2.GE.ITEMP1)GOTO 320
         ITEMP1=ITEMP2
         MINPTO=NPTTO
         ITEMP3=N
         L=LTEMP
  320 C O N T I N U E
      IF(MINPTO.GT.0)GOTO 324
C   FOLLOWING TO STATEMENT 324 EXECUTED FOR UNDEFINED RENTACAR FACILITY.
      L=0
      DO 322 N=I,J
         L=L+1
         K=IMAXBH(MH9(N,4))
         IF(K.GT.0)GOTO 323
  322 C O N T I N U E
      IF(NRCRSW.EQ.1)GOTO 9 9 9 9
      NRCRSW=1
      WRITE(6,1019)
      GOTO 9 9 9 9
  323 NPTTO=IMAXBH(MH9(N,3))
      ITEMP3=N
      WRITE(6,1018)IV3,K
      NERCNT=NERCNT+1
      IF(NERCNT.EQ.ERRORS)GOTO 999
      IV3=K
      MINPTO = NPTTO
      GOTO 325
  324 NPTTO=MINPTO
  325 ASSIGN 326 TO NEXT
      GOTO 950
  326 M=RCRQS+L-1
      CALL ASSIGN( 2,MINPTO,PH, 5,M,PH, 7,ITEMP3,PH, 11,11,PB )
      GOTO 9 9 9 9
C
C
    7        C    O    N    T    I    N    U    E
C
C...E X I T
C
C
C                  IVALUE(2) = CURRENT LOCATION - PH2
C                  IVALUE(3) = CURRENT PROCESS - PB11
C                  IVALUE(4) = NEXT ADDRESS - FN*PB1
C                  IVALUE(5) = MH9 ROW OF LAST FACILITY - PH7
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
      IV4=IVALUE(4)
      IV5=IVALUE(5)
C   SCAN VALID FACILITY TYPES TO EXIT TO.
      IF(IV4.EQ.DPLCO.OR.IV4.EQ.CGTRO.OR.IV4.EQ.GRTOO)GOTO 510
      I=PVAL(PB,1)
      WRITE(6,1008)IVALUE(4),I
      NERCNT=NERCNT+1
```

```
      IF(NERCNT.EQ.ERRORS)GOTO 999
      GOTO 9 9 9 9 9
C   EXIT TO DEPLANING CURB.   CHECK FOR FACILITY LEAVING FROM.
  510 IF(IV3.EQ.1)GOTO 520
C  WW CAN LEAVE FROM SECURITY
      IF (IV3.EQ.3) GO TO 535
      IF(IV3.EQ.4)GOTO 515
      IF(IV3.EQ.5)GOTO 525
      IF(IV3.EQ.11)GOTO 530
      WRITE(6,1009)FACTYP(IV3)
      GOTO 9 9 9 9 9
C   NOTE: COMMONALITY IN FOLLOWING CODE BLOCKS TO PERMIT TAILORING FOR
C         A SPECIFIC INSTALLATION.
C   BAG CLAIM - DEPLANING CURB
  515 J=IMAXBH(MH9(IV5,3))
      NPTTO=IMAXBH(MH3(J,3))
      ASSIGN 516 TO NEXT
      GOTO 950
  516 CALL ASSIGN( 2,NPTTO,PH )
      GOTO 9 9 9 9 9
C   GATE - DEPLANING CURB
  520 J=IMAXBH(MH9(IV5,3))
      NPTTO=IMAXBH(MH3(J,3))
      ASSIGN 521 TO NEXT
      GOTO 950
  521 CALL ASSIGN( 2,NPTTO,PH )
      GOTO 9 9 9 9 9
C   CUSTOMS - DEPLANING CURB
  525 J=IMAXBH(MH9(IV5,3))
      NPTTO=IMAXBH(MH3(J,3))
      ASSIGN 526 TO NEXT
      GOTO 950
  526 CALL ASSIGN( 2,NPTTO,PH )
      GOTO 9 9 9 9 9
C   CAR RENTAL - DEPLANING CURB
  530 J=IMAXBH(MH9(IV5,3))
      NPTTO=IMAXBH(MH3(J,3))
      ASSIGN 531 TO NEXT
      GOTO 950
  531 CALL ASSIGN( 2,NPTTO,PH )
      GOTO 9 9 9 9 9
C   SECURITY - DEPLANING CURB
  535 J = IMAXBH(MH9(IV5,3))
      NPTTO = IMAXBH(MH3(J,3))
      ASSIGN 536 TO NEXT
      GO TO 950
  536 CALL ASSIGN (2,NPTTO,PH)
      GO TO 9 9 9 9 9
C
C
    8        C    O    N    T    I    N    U    E
C
C...I M M I G R A T I O N
C
C                 IVALUE(2) = CURRENT LOCATION - PH2
C                 IVALUE(3) = GATE NUMBER - MH1(PH1,9)
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
      L=IMAXBH(MH9(IV3,5))
C   TEST FOR GATE'S DESIGNATED IMMIGRATION FACILITY
```

```
      IF(L.GT.0)GOTO 335
      IF(NOIMMI.GT.0)GOTO 331
      WRITE(6,1010)
      NERCNT=NERCNT+1
      IF(NERCNT.EQ.ERRORS)GOTO 999
      GOTO 9 9 9 9 9
C   NO IMMIGRATION AREA SPECIFIED FOR GATE.   FIND ANY IMMIGRATION AREA.
  331 J=INDEXF(13)
      K=J+NOIMMI
      J=J+1
      DO 332 N=J,K
         L=L+1
         IF(IMAXBH(MH9(N,3)).GT.0)GOTO 334
  332 C O N T I N U E
  334 WRITE(6,1011)IV3,L
      NERCNT=NERCNT+1
      IF(NERCNT.EQ.ERRORS)GOTO 999
  335 J=INDEXF(13)+L
      NPTTO=IMAXBH(MH9(J,3))
      ASSIGN 338 TO NEXT
      GOTO 950
  338 M=IMMQS+L-1
      CALL ASSIGN( 2,NPTTO,PH, 5,M,PH, 7,J,PH, 11,13,PB, 8,J,PH )
      GOTO 9 9 9 9 9
C
C
    9        C     O     N     T     I     N     U     E
C
C...D E P L A N I N G   C U R B   ( P A X )
C
C                 IVALUE(2) = CURRENT LOCATION - PH2
C                 IVALUE(3) = LAST FACILITY TYPE (OTHER THAN EXIT) - PB1
C                 IVALUE(4) = LAST MH9 ROW (OTHER THAN EXIT) - PH7
C                 IVALUE(5) = MH1 ROW - PH1
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
      IV5=IVALUE(5)
      I = IMAXBH(MH1(IV5,3))
C   SCAN FOR VALID FACILITY TYPES COMING FROM
      IF(IV3.EQ.1)GOTO 600
      IF(IV3.EQ.4)GOTO 605
      IF(IV3.EQ.5)GOTO 610
      IF(IV3.EQ.11)GOTO 615
      IF (IV3.EQ.14) GO TO 620
      I=PVAL(PB,1)
      WRITE(6,1012)FACTYP(IV3),I
      NERCNT=NERCNT+1
      IF(NERCNT.EQ.ERRORS)GOTO 999
      GOTO 9 9 9 9 9
C   COMING DIRECTLY FROM GATE - FIND ASSIGNED BAG CLAIM AREA FOR FLIGHT
  600 I=IMAXBH(MH1(IV5,12))+INDEXF(4)
      ITEMP1=IMAXBH(MH9(I,4))
      GOTO 690
C   COMING FROM BAG CLAIM
  605 I=IVALUE(4)
      ITEMP1=IMAXBH(MH9(I,4))
      GOTO 690
C   COMING FROM CUSTOMS
  610 I=IVALUE(4)
      ITEMP1=IMAXBH(MH9(I,4))
```

```
      GOTO 690
C   COMING FROM RENTACAR
  615 I=IVALUE(4)
      ITEMP1=IMAXBH(MH9(I,5))
      GOTO 690
C  COMING FROM CHECKIN--DEPLANING LOBBY PAX TO ENPLANING CURB
  620 I = IMAXBH(MH1(IV5,3))
      I = IMAXBH(MH2(I,1))
      J = I+INDEXF(8)
      GO TO 692
C   DETERMINE DELPANING CURB AREA
  690 J=ITEMP1+INDEXF(12)
  692 NPTTO=IMAXBH(MH9(J,3))
      ASSIGN 691 TO NEXT
      GOTO 950
  691 CALL ASSIGN( 2,NPTTO,PH, 7,J,PH, 11,12,PB )
      GOTO 9 9 9 9 9
C
C
   10         C    O    N    T    I    N    U    E
C
C...D E P L A N I N G   C U R B   ( C A R S   &   G R E E T E R S )
C
C                    IVALUE(2) = AIRLINE
C                    IVALUE(3) = MH1 ROW - PH1
C                    IVALUE(4) = NUMBER OF BAGS (INDICATES DEPL OR ENPL CB)
C                    IVALUE(5) = 1 IF GREETER (RECIRCULATED AND PARKED)
C
      IV2=IVALUE(2)
      IV3=IVALUE(3)
      IV4 = IVALUE(4)
      IF (IV4.NE.0) GO TO 700
C
C  USING ENPLANING CURB
C
      M = IMAXBH(MH2(IV2,1))
      IF (IVALUE(5).EQ.1) GO TO 716
C  CURB SEARCH SCHEME FOR OPEN CURB OR DP SLOTS
      DO 713 K=1,10
         L = IEPSCH(K,M)
C  IGNORE FACILITY NUMBERS > NOENPL
         IF (L.GT.NOENPL) GO TO 713
         ITEMP1 = INDEXF(8)+L
C  TEST FOR DUMMY FACILITY
         IF (IMAXBH(MH9(ITEMP1,3)).EQ.0) GO TO 713
         J = EPCBS+L-1
         ITEMP3 = 11*(J-1)+2
         IF (ISTO(ITEMP3).EQ.0) GO TO 714
C  CAR GETS CURB SLOT
         CALL ASSIGN(6,J,PH, 10,1,PB)
         GO TO 9 9 9 9 9
  714    J = EPDPS+L-1
         ITEMP3 = 11*(J-1)+2
         IF (ISTO(ITEMP3).EQ.0) GO TO 713
C  CAR GETS DP SLOT
         CALL ASSIGN(6,J,PH, 10,2,PB)
         GO TO 9 9 9 9 9
  713 CONTINUE
      L = M
      J = EPQCS+L-1
      ITEMP3 = 11*(J-1)+2
```

```
      IF(IV3.EQ.5)GOTO 801
C   PVTCAR OR TAXI - GET ENPLANING CURB FAC NO FOR AIRLINE
  801 DO 800 K=1,10
C   POINT TO CURB SEARCH SCHEME
          L=IEPSCH(K.J)
C   IGNORE FACILITY NUMBERS GT NOENPL
          IF(L.GT.NOENPL)GOTO 800
          ITEMP1=INDEXF(8)+L
C       TEST FOR DUMMY FACILITY
          IF(IMAXBH(MH9(ITEMP1,3)).EQ.0)GOTO 800
          M=EPCBS+L-1
          ITEMP3=11*(M-1)+2
          IF (ISTO(ITEMP3).EQ.0) GO TO 804
C  CAR GETS CURB SLOT
          CALL ASSIGN(6,M,PH, 10,1,PB)
          GO TO 803
  804     M = EPDPS+L-1
          ITEMP3 = 11*(M-1)+2
          IF (ISTO(ITEMP3).EQ.0) GO TO 800
C  CAR GETS DP SLOT
          CALL ASSIGN(6,M,PH, 10,2,PB)
          GO TO 803
  800 CONTINUE
      L = J
      ITEMP1 = INDEXF(8)+L
      M = EPQCS+L-1
      ITEMP3 = 11*(M-1)+2
      IF (ISTO(ITEMP3).EQ.0) GO TO 805
C  CAR GETS QUEUE SLOT
      CALL ASSIGN(6,M,PH, 10,3,PB)
      GO TO 803
C  CAR MUST RECIRCULATE
  805 CALL ASSIGN(5,0,PH, 6,0,PH, 10,4,PB)
      GO TO 9 9 9 9 9
C   M=ENPLCURB STO, ITEMP1=MH9ROW, ITEMP3=CAR CURB STO
  803 NPTTO=IMAXBH(MH9(ITEMP1,3))
      CALL ASSIGN( 2,NPTTO,PH, 7,ITEMP1,PH )
      GOTO 9 9 9 9 9
C   BUS/LIMO
  808 ITEMP2=IMAXBH(MH2(IV2,3))
      IF(ITEMP2.GT.0)GOTO 809
      ITEMP2=IMAXBH(MH2(IV2,1))
  809 ITEMP1=INDEXF(8)+ITEMP2
      NPTTO=IMAXBH(MH9(ITEMP1,3))
      CALL ASSIGN( 2,NPTTO,PH, 7,ITEMP1,PH )
      GOTO 9 9 9 9 9
C
C
   12         C    O    N    T    I    N    U    E
C
C...E N T R A N C E
C
C                 IVALUE(2) = CURRENT LOCATION - PH2
C
      NPTFM=IVALUE(2)
      NPTTO=IMAXBH(MH3(NPTFM,4))
      ASSIGN 813 TO NEXT
      GOTO 950
  813 CALL ASSIGN( 2,NPTTO,PH )
      GOTO 9 9 9 9 9
C
```

```
C
   13        C   O   N   T   I   N   U   E
C
C...T I C K E T I N G   &   C H E C K I N    ( A L L )
C
C                   IVALUE(2) = CURRENT LOCATION - PH2
C                   IVALUE(3) = AIR LINE - MH1(PH1,3)
C                   IVALUE(4) = TICKETED/NOT TICKETED (0,1) - PB9
C                   IVALUE(5) = RANDOM NO. FOR FRACTIONAL TRANSFER
C                   IVALUE(6) = NUMBER OF PAX
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
C   IF TERMINATING (PASSING THROUGH LOBBY), BRANCH TO FULL-SERVICE
      IF (PVAL(PB,8).EQ.1) GO TO 844
C   IF GREETER OR GREETED, BRANCH TO FULL-SERVICE TICKETING
      IF (IVALUE(6).EQ.0.OR.PVAL(PB,12).EQ.3) GO TO 844
C   IF PAX NOT PRETICKETED .OR. RANDOM NO .GT. EXPCHK .....
C   ... BRANCH TO FULL SERVICE SECTION.
      IF(IVALUE(4).EQ.1.OR.IVALUE(5).GT.IMAXBH(MH2(IV3,2)))GOTO 844
      GOTO 850
C
C   FULL SERVICE FACILITY
C
  844 J=INDEXF(14)
      K=J+NOTICK
      J=J+1
      L=0
      DO 845 I=J,K
         L=L+1
         IF(IMAXBH(MH9(I,4)).EQ.IV3)GOTO 848
  845 C O N T I N U E
C   FOLLOWING EXECUTED FOR UNDEFINED FACILITY
      IF(NOTICK.GT.0)GOTO 847
      WRITE(6,1028)
      GOTO 999
  847 L=1
      I=INDEXF(14)+1
      N=IMAXBH(MH9(I,4))
      WRITE(6,1027)IV3,N
      NERCNT=NERCNT+1
      IF(NERCNT.EQ.ERRORS)GOTO 999
  848 M=TICQS+L-1
      ITEMP1=CHEK3
      N = 14
      GOTO 857
C
C   EXPRESS CHECKIN FACILITY
C
  850 J=INDEXF(2)
      K=J+NOCHEC
      J=J+1
      L=0
      DO 851 I=J,K
         L=L+1
         IF(IMAXBH(MH9(I,4)).EQ.IV3)GOTO 853
  851 C O N T I N U E
C   FOLLOWING CODE EXECUTED FOR UNDEFINED FACILITY
      J=INDEXF(14)
      K=J+NOTICK
      J=J+1
```

```
      L=0
C   SEARCH FOR FULL SERVICE FACILITY FOR THIS AIRLINE
      DO 858 I=J,K
         L=L+1
         IF(IMAXBH(MH9(I,4)).EQ.IV3)GOTO 859
  858 C O N T I N U E
C   USE ANY FULL SERVICE FACILITY
      IF(NOTICK.GT.0)GOTO 852
      WRITE(6,1028)
      GOTO 999
  852 I=INDEXF(14)+1
      N=IMAXBH(MH9(I,4))
      WRITE(6,1029)IV3,N
      NERCNT=NERCNT+1
      L=1
      IF(NERCNT.EQ.ERRORS)GOTO 999
  859 M=TICQS+L-1
      ITEMP1=CHEK3
      N=14
      GOTO 857
  853 M=CHKQS-1+L
      N=2
      ITEMP1=CHEK2
      GOTO 857
  857 NPTTO=IMAXBH(MH9(I,3))
      ASSIGN 856 TO NEXT
      GOTO 950
  856 CALL ASSIGN( 2,NPTTO,PH, 4,ITEMP1,PH, 5,M,PH, 7,I,PH, 11,N,PB )
      GOTO 9 9 9 9
C
C
   14         C    O    N    T    I    N    U    E
C
C...S E C U R I T Y
C
C                IVALUE(2) = CURRENT LOCATION - PH2
C                IVALUE(3) = GATE - MH1(PH1,9)
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
C   DETERMINE SECURITY FACILITY ASSIGNED TO THIS GATE
      I=IMAXBH(MH9(IV3,4))
      IF(I.GT.0)GOTO 860
      WRITE(6,1013)IV3
      IMAXBH(MH9(IV3,4))=1
      I=1
C   DETERMINE LOCATION OF SECURITY POINT.
  860 J=INDEXF(3)+I
      M=SECQS+I-1
      NPTTO=IMAXBH(MH9(J,3))
C   NOTE: MODIFY NEXT CALCULATION TO REFLECT EARLY PASSENGERS WAITING
C         UNTIL CLOSER TO FLIGHT TIME TO PROCEED TO GATE.   PASS CURRENT
C         TIME (C1) AND FLIGHT TIME (MH1(PH1,6)) VIA IVALUE LIST.
      ASSIGN 861 TO NEXT
      GOTO 950
  861 CALL ASSIGN( 2,NPTTO,PH, 5,M,PH, 7,J,PH, 11,3,PB )
      GOTO 9 9 9 9
C
C
   15         C    O    N    T    I    N    U    E
C
```

```
C...G A T E   ( E N P L A N I N G   P A X )
C
C                 IVALUE(2) = CURRENT LOCATION - PH2
C                 IVALUE(3) = GATE - MH1(PH1,9)
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
      NPTTO=IMAXBH(MH9(IV3,3))
      IF(NPTTO.GT.0)GOTO 873
      DO 871 I=1,NOGATE
         IF(IMAXBH(MH9(I,3)).NE.0)GOTO 872
  871 C O N T I N U E
  872 J=PVAL(PH,1)
      IMAXBH(MH1(J,9))=I
      WRITE(6,1014)IV3,IMAXBH(MH1(J,2)),I
      IV3=I
      NPTTO=IMAXBH(MH9(IV3,3))
  873 ASSIGN 874 TO NEXT
      GOTO 950
  874 M=GAQSL+IV3-1
      CALL ASSIGN( 2,NPTTO,PH, 5,M,PH, 7,IV3,PH, 11,1,PB )
      GOTO 9 9 9 9 9
C
C
   16       C    O    N    T    I    N    U    E
C
C...P A R K I N G   ( P A X )
C
C        NOTE: UNLIKE THE CODE FOR MOST FACILITY TYPES, THE FORTRAN
C              CODE FOR "PARKING" MAY BE CALLED FROM A VARIETY OF
C              POINTS WITHIN THE GPSS PORTION OF THIS MODEL.
C
C                 IVALUE(2) = CURRENT LOCATION - PH2
C                 IVALUE(3) = TRANSPORTATION MODE - PB6
C                 IVALUE(4) = DEPLANING/ENPLANING (0/1)
C                 IVALUE(5) = CAR RENTAL AGENCY (PB10) WHEN IVALUE(3)=3
C                 IVALUE(6) = 1 TO GET LOT NUMBER ONLY
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
      IV4=IVALUE(4)
      IV5=IVALUE(5)
      IV6 = IVALUE(6)
      IF(IV4.EQ.1)GOTO 720
C   TESTS FOR DEPLANING PAX
      IF (IV3.EQ.1) GO TO 728
      IF(IV3.EQ.2)GOTO 728
      IF(IV3.EQ.3)GOTO 722
      GOTO 721
C   TESTS FOR ENPLANING PAX
  720 IF(IV3.EQ.2)GOTO 728
      IF(IV3.EQ.3)GOTO 722
      IF (IV3.EQ.1) GO TO 728
C   ERROR CONDITION
  721 I=PVAL(PH,4)
      WRITE(6,1015)NPTFM,I,IV4,IV3
      NERCNT=NERCNT+1
      IF(NERCNT.EQ.ERRORS)GOTO 999
      GOTO 9 9 9 9 9
C   DEPLANING PAX - RENTAL CAR
C   ENPLANING PAX - RENTAL CAR
```

```
C   DETERMINE IF AGENCY HAS SPECIAL LOT
  722 I=INDEXF(11)
      J=I+NORENT
      I=I+1
      DO 725 N=I,J
         IF(IMAXBH(MH9(N,4)).NE.IV5)GOTO 725
         L=IMAXBH(MH9(N,5))
         IF(L.GT.1) GOTO 723
  725 C O N T I N U E
C   DEPLANING PAX - SELF
C   ENPLANING PAX - SELF
C   GENERAL LOT
  728 LOTNO = PVAL(PB,14)
      IF (LOTNO.EQ.0) LOTNO = 1
C  INSERT ASSIGNMENT OF MULTIPLE LOTS HERE
      N=INDEXF(10)+LOTNO
      M=PARQS+LOTNO-1
      IF (IV6.NE.1) GO TO 724
      CALL ASSIGN(14,LOTNO,PB)
      GO TO 9 9 9 9 9
C   SPECIAL LOT
  723 N=INDEXF(10)+L
      M=PARQS+L-1
  724 NPTTO=IMAXBH(MH9(N,3))
      IF (NPTFH.EQ.0) GO TO 727
      ASSIGN 727 TO NEXT
      GOTO 950
  727 CALL ASSIGN( 2,NPTTO,PH, 5,M,PH, 7,N,PH, 11,10,PB, 14,LOTNO,PB)
      GOTO 9 9 9 9 9
C
C
   17         C    O    N    T    I    N    U    E
C
C...T R A N S F E R   P A X
C
C                   IVALUE(2) = SWITCH: 1=TRANSFER, 2=TRANSIT
C                   IVALUE(3) = RANDOM NO FOR FLT SELECTION (TRANSFER)
C                             = ARRIVING FLIGHT NUMBER PH1 (TRANSIT)
C                   IVALUE(4) = DOM/COM/INT PAX (1/2/3) - PB3 (TRANSFER)
C                   IVALUE(5) = GATE NO. - PH5
C
      M=IVALUE(5)
      ITEMP3=IMAXBH(MH9(M,4))
      IF(ITEMP3.GT.0) GO TO 827
      WRITE(6,1013) M
      IMAXBH(MH9(M,4))=1
      ITEMP3=1
  827 IV2=IVALUE(2)
      GO TO (821,822),IV2
C
C  TRANSFER PAX
C
  821 IF(NOFXFR.GT.0)GOTO 824
      K=PVAL(PB,5)
      IMAXBH(MH11(ITEMP3))=IMAXBH(MH11(ITEMP3))+K
      ISAVEH(XFRXH)=ISAVEH(XFRXH)+1
      CALL ASSIGN( 4,TRX99,PH, 8,CTRL1,PH )
      GOTO 9 9 9 9 9
  824 CALL ASSIGN( 8,CTRLO,PH )
C   RANDOMLY CHOSE FLIGHT
      N=MOD(IVALUE(3),NOFXFR)+1
```

```
      I=IMAXBH(MH5(N))
      K=MH1(I,11)
      IMAXBH(K)=IMAXBH(K)-1
C   WHEN ALL TRANSFER PAX FOR FLT ASSIGNED, DELETE FLT FROM TABLE.
      IF(IMAXBH(K).GT.0)GOTO 820
      DO 823 L=N,NOFXFR
         ITEMP1=MH5(L)
         ITEMP2=ITEMP1+1
         IMAXBH(ITEMP1)=IMAXBH(ITEMP2)
  823 C O N T I N U E
      NOFXFR=NOFXFR-1
  820 CALL ASSIGN( 1,I,PH )
      GOTO 9 9 9 9 9
C
C  TRANSIT PAX
C
  822 K = IVALUE(3)
C  FIND GATE OF ARRIVING FLIGHT
      IGAT = IMAXBH(MH1(K,9))
      K = K+1
C  FIND NEXT DEPARTURE AT SAME GATE
      DO 826 I=K,999
         IF (IMAXBH(MH1(I,1))) 818,826,819
  819    IF (IMAXBH(MH1(I,9)).EQ.IGAT) GO TO 817
  826 CONTINUE
C  NO NEXT DEPARTURE IN TABLE
  818 K = PVAL(PB,5)
      IMAXBH(MH11(ITEMP3))=IMAXBH(MH11(ITEMP3))+K
      ISAVEH(XFRXH) = ISAVEH(XFRXH)+1
      CALL ASSIGN (4,TRX99,PH, 8,CTRL1,PH)
C  XAC WILL BE TERMINATED
      GO TO 9 9 9 9 9
  817 CALL ASSIGN (1,I,PH, 8,CTRL0,PH)
      GO TO 9 9 9 9 9
C
C
   18         C    O    N    T    I    N    U    E
C
C...T R A N S F E R   F L I G H T S
C
C                    IVALUE(2) = MH1 ROW NO - PH1
C                    IVALUE(3) = INIT./DELETE/ADD/TICK CNTER PT NO 0/1/2/3
C
      IV2=IVALUE(2)
      IV3=IVALUE(3)
      IF(IV3.EQ.1)GOTO 832
      IF(IV3.EQ.2)GOTO 830
      IF(IV3.EQ.3) GO TO 836
C   INITIALIZE TABLE
      DO 834 I=1,999
C       TEST: END_OF_TABLE/ARV_FLT/DEP_FLT
         IF(IMAXBH(MH1(I,1)))835,834,833
  833    ITEMP1=IMAXBH(MH1(I,6))*60
         IF(ITEMP1.GT.ISAVEH(XFAXH))GOTO 835
         IF(ITEMP1.LT.ISAVEH(XFDXH))GOTO 834
         IF(IMAXBH(MH1(I,11)).EQ.0)GOTO 834
         NOFXFR=NOFXFR+1
         IMAXBH(MH5(NOFXFR))=I
  834 C O N T I N U E
  835 CALL ASSIGN( 1,I,PH )
      GOTO 9 9 9 9 9
```

```
C   DELETE FLIGHT FROM TABLE MH5
  832 IF(IMAXBH(MH5(1)).NE.IV2)GOTO 9 9 9 9 9
      DO 829 I=1,NOFXFR
         ITEMP1=MH5(I)
         ITEMP2=ITEMP1+1
         IMAXBH(ITEMP1)=IMAXBH(ITEMP2)
  829 C O N T I N U E
      NOFXFR=NOFXFR-1
      GOTO 9 9 9 9 9
C   ADD FLIGHT TO TABLE MH5
  830 IF(NOFXFR.EQ.100)GOTO 831
      NOFXFR=NOFXFR+1
      IMAXBH(MH5(NOFXFR))=IV2
      GOTO 9 9 9 9 9
C   ERROR - TABLE OVERFLOW.
  831 WRITE(6,1023)IV2
      GOTO 9 9 9 9 9
C   FIND TICKET COUNTER FOR CORRECT AIRLINE FOR TRANSFER PAX
  836 IAIRLN=IMAXBH(MH1(IV2,3))
      IROWNO=IMAXBH(MH8(14,2))
      INUMTC=IMAXBH(MH8(14,1))
      ITEMP1=IROWNO+1
      ITEMP2=IROWNO+INUMTC
      DO 837 I=ITEMP1,ITEMP2
      IF(IMAXBH(MH9(I,4)).EQ.IAIRLN) GO TO 838
  837 CONTINUE
      I=ITEMP1
      ITEMP2=IMAXBH(MH9(I,4))
      WRITE(6,1029) IAIRLN,ITEMP2
      WRITE(6,1033) IV2,IV3
  838 IPTNO=IMAXBH(MH9(I,3))
      CALL ASSIGN(2,IPTNO,PH)
      GO TO 9 9 9 9 9
C
C
   19         C    O    N    T    I    N    U    E
C
C...M I S C E L L A N E O U S   G P S S   E R R O R   C O N D I T I O N
C
C                  CALLED FROM GPSS TO RECORD A VARIETY OF ERROR CONDITIO
C                  CALLING XAC'S FOUND ON USER CHAIN "ERROR" AT END OF RU
C
      IV2=IVALUE(2)
      GOTO(901,902,903,904,905,906,907,908,909,910),IV2
C   NO VEHICLE-PAX MATCH AT DEPLANING CURB
  901 WRITE(6,1016)IVALUE(3)
      GOTO 9 9 9 9 9
C   PAX ENTERED DEPLCURB LOGIC WITH GR TX CODE LOGIC NOT CODED TO HANDLE
  902 WRITE(6,1017)IVALUE(3),IVALUE(4)
      GOTO 9 9 9 9 9
  903 CONTINUE
  904 CONTINUE
  905 CONTINUE
  906 CONTINUE
  907 CONTINUE
  908 CONTINUE
  909 CONTINUE
  910 CONTINUE
      GOTO 9 9 9 9 9
C
C
```

```
   20       C   O   N   T   I   N   U   E
C
C...F O R M A T T E D   R E P O R T S
C
      C1=IVALUE(2)
C   SEARCH FACILITY TYPES.
      DO 450 I=1,20
         NSWTCH=0
         K=IMAXBH(MH8(I,1))
C       BRANCH IF NO FACILITIES FOR TYPE "I".
         IF(K.EQ.0)GOTO 450
C       SET DO-LOOP VARIABLES FOR SCAN OF FACILITY TABLE (MH9).
         J=IMAXBH(MH8(I,2))
         K=K+J
         J=J+1
C       BRANCH TO APPROPRIATE HEADER FOR:
C                         GATES           CHECKIN/TICKETING
C                         CUSTOMS         CAR RENTAL
C                         SECURITY        IMMIGRATION
C       SKIP OTHER FACILITY TYPES.
      GOTO( 400,400,400,450,400,450,450,450,450,450,
     *      400,450,400,400,450,450,450,450,450,450), I
  400 IF(NTLINS.GT.0)WRITE(6,1050)((ITITLE(II,JJ),II=1,64),JJ=1,NTLINS)
         GOTO( 401,402,403,450,405,450,450,450,450,450,
     *         411,450,413,414,450,450,450,450,450,450), I
C       BOARDING GATES
  401    WRITE(6,1051)
         GOTO 430
C       CHECKIN(EXPRESS)
  402    WRITE(6,1052)
         GOTO 430
C       SECURITY
  403    WRITE(6,1053)
         GOTO 430
C       CUSTOMS
  405    WRITE(6,1055)
         GOTO 430
C       CAR RENTAL
  411    WRITE(6,1061)
      GOTO 430
C       IMMIGRATION
  413    WRITE(6,1063)
         GOTO 430
C       TICKETS&CHECKIN
  414    WRITE(6,1064)
         GOTO 430
C       COMPLETE HEADING.   THEN CHECK EACH FACILITY OF TYPE "I".
  430    WRITE(6,1092)
         WRITE(6,1094)
         WRITE(6,1096)
         NCOUNT=11+NTLINS
         ITEMP1=FACQSX(I)
         IQUER=4*(ITEMP1-1)
         IQUEI=IQUER+IQUER
         ISTOX=11*(ITEMP1-1)
         ITEMP1=ITEMP1-FACQSX(I)+1
         DO 455 N=J,K
C          CHECK FOR DUMMY FACILITY.
            IF(IMAXBH(MH9(N,3)).EQ.0)GOTO 448
            NCOUNT=NCOUNT+2
C          CHECK FOR FULL PAGE (55 LINES).
```

```
      IF(NCOUNT.LE.55)GOTO 445
      WRITE(6,1078)
  IF(NTLINS.GT.0)WRITE(6,1050)((ITITLE(II,JJ),II=1,64),JJ=1,NTLINS)
      GOTO( 421,422,423,450,425,450,450,450,450,450,
     *      431,450,433,434,425,450,450,450,450,450),  I
C     BOARDING GATES
  421 WRITE(6,1051)
      GOTO 443
C     CHECKIN(EXPRESS)
  422 WRITE(6,1052)
      GOTO 443
C     SECURITY
  423 WRITE(6,1053)
      GOTO 443
C     CUSTOMS
  425 WRITE(6,1055)
      GOTO 443
C     CAR RENTAL
  431 WRITE(6,1061)
      GOTO 443
C     IMMIGRATION
  433 WRITE(6,1063)
      GOTO 443
C     TICKETS&CHECKIN
  434 WRITE(6,1064)
      GOTO 443
  443 NCOUNT=11+NTLINS
      WRITE(6,1092)
      WRITE(6,1094)
      WRITE(6,1096)
  445 ITEMP2=ISTO(ISTOX+1)+ISTO(ISTOX+2)
C   CHECK FOR UNDEFINED NUMBER OF AGENTS.   1000 ARBITRARY NUMBER.
      IF(ITEMP2.GT.1000)NSWTCH=1
      ITEMP3=ISTO(ISTOX+6)*SCALE
      IF(ITEMP3.GT.0)GOTO 444
      ITEMP4=0
      XTEMP5=0.0
      ITMP6M=0
      ITMP6S=0
      GOTO 446
  444 ITEMP4=ISTO(ISTOX+7)
      XTEMP5=FSTO(ISTOX+3)/C1
      ITEMP6=FSTO(ISTOX+3)/ITEMP3
      ITMP6M=ITEMP6/60
      ITMP6S=MOD(ITEMP6,60)
  446 ITEMP7=IQUE(IQUEI+2)*SCALE
      IF(ITEMP7.GT.0)GOTO 447
      ITEMP8=0
      XTEMP9=0.0
      ITM10M=0
      ITM10S=0
      GOTO 449
  447 ITEMP8=IQUE(IQUEI+7)*SCALE
      XTEMP9=FQUE(IQUER+2)*SCALE/C1
      ITMP10=FQUE(IQUER+2)*SCALE/ITEMP7
      ITM10M=ITMP10/60
      ITM10S=MOD(ITMP10,60)
  449 WRITE(6,1075)ITEMP1,ITEMP2,ITEMP3,ITEMP4,XTEMP5,ITMP6M,
     *             ITMP6S,ITEMP7,ITEMP8,XTEMP9,ITM10M,ITM10S
  448 ITEMP1=ITEMP1+1
      IQUER=IQUER+4
```

```
          IQUEI=IQUEI+8
          ISTOX=ISTOX+11
  455   C O N T I N U E
        WRITE(6,1078)
C      TEST FOR UNDEFINED NO. OF AGENTS.
        IF(NSWTCH.EQ.1)WRITE(6,1079)
  450 C O N T I N U E
C
      GOTO 9 9 9 9 9
C
C
   21         C     O     N     T     I     N     U     E
C
C...C L O C K   U P D A T E
C
C                   IVALUE(2) = TIME INCREMENT (SECONDS)
C
      ITEMP1=ISAVEH(CLKXH)+IVALUE(2)/60
      IF(MOD(ITEMP1,100).GE.60)ITEMP1=ITEMP1+40
      ISAVEH(CLKXH)=ITEMP1
      GOTO 9 9 9 9 9
C
C
   22         C     O     N     T     I     N     U     E
C
C...S N A P S H O T S
C
C
C   STORAGE OUTPUT FLOW
C
      NSWTCH=0
      ITEMP1=ISAVEH(CLKXH)
      IF(LINSNP.LT.50) GO TO 653
      NSWTCH=1
      LINSNP=NTLINS
      IF(NTLINS.GT.0)WRITE(12,1050)((ITITLE(I,J),I=1,64),J=1,NTLINS)
      WRITE(12,1074)
      WRITE(12,1082)
      WRITE(12,1076)
  653 DO 654 I=1,20
        ITEMPA(I)=ISAVEH(I)*SCALE
  654 C O N T I N U E
      WRITE(12,1077)ITEMP1,(ITEMPA(I),I=1,24)
      IF(LINSNX.LT.50) GO TO 960
      LINSNX=NTLINS
      IF(NTLINS.GT.0)WRITE(13,1050)((ITITLE(II,JJ),II=1,64),JJ=1,NTLINS)
      WRITE(13,1070)
      WRITE(13,1082)
      WRITE(13,1076)
  960 LINSNX=LINSNX+1
      DO 660 IR=1,24
      ISTRNO=GPSTO(IR)
      IF(ISTRNO.EQ.0) GO TO 965
      JENTCT=11*(ISTRNO-1)+6
      JCRCON=11*(ISTRNO-1)+1
      XENTCT=ISTO(JENTCT)
      XCRCON=ISTO(JCRCON)
      FLOW=((XENTCT-ENTRCT(IR))-(XCRCON-CRCON(IR)))*SCALE
      ENTRCT(IR)=XENTCT
      CRCON(IR)=XCRCON
       TSSOUT(1)=ITEMP1
```

```
      TSSOUT(IR+1)=FLOW
C
C   QUEUE LENGTHS
C
  965 ITQUE=GPQUE(IR)
      IF(ITQUE.EQ.0) GO TO 967
      JQUE=8*(ITQUE-1)+6
       TSQUE(1)=ITEMP1
       TSQUE(IR+1)=IQUE(JQUE)*SCALE
C
C   HALF-WORD SAVEVALUES
C
  967 ITHLF=GPHALF(IR)
      IF(ITHLF.EQ.0) GO TO 660
      ISHLF=ISAVEH(ITHLF)
      FLOW=(ISHLF-JTHLF(IR))*SCALE
       TSHALF(1)=ITEMP1
       TSHALF(IR+1)=FLOW
      JTHLF(IR)=ISHLF
  660 CONTINUE
      DO 969 IL=1,7
       JSECFL=IMAXBH(MH12(IL))
       TSFLOW(1)=ITEMP1
       TSFLOW(IL+1)=(JSECFL-ISECFL(IL))*SCALE
  969 ISECFL(IL)=JSECFL
      DO 973 IT=1,15
        JTCKFL=IMAXBH(MH13(IT))
        TTFLOW(1)=ITEMP1
       TTFLOW(IT+1)=(JTCKFL-ITCKFL(IT))*SCALE
  973 ITCKFL(IT)=JTCKFL
        WRITE(13,1077) (TSSOUT(IP),IP=1,25)
        WRITE(13,1095) (TSQUE(IP),IP=2,25)
        WRITE(13,1095) (TSHALF(IP),IP=2,25)
        WRITE(13,1095) (TSFLOW(IL),IL=2,8)
        WRITE(13,1095) (TTFLOW(IP),IP=2,16)
       WRITE(14,1097) (TSSOUT(IP),IP=1,25),
     *(TSHALF(IP),IP=2,25),
     *(TSQUE(IP),IP=2,25),
     *(TSFLOW(IL),IL=2,8),
     *(TTFLOW(IT),IT=2,16)
      GOTO 9 9 9 9
C
C
   23         C    O    N    T    I    N    U    E
C
C...C H A N G E   C A R D   P R O C E S S I N G
C
C                 IVALUE(2) = SWITCH, =1 TO READ CARD
C                                     =2 TO LOWER STORAGE
C                 IVALUE(3) = STORAGE NUMBER FOR LOWERING
C                 IVALUE(4) = DESIRED STORAGE CAPACITY
C
      IF (IVALUE(2).EQ.2) GO TO 590
C
C  CHANGE CARD PROCESSING
C
C  BRANCH IF FIRST ENTRY
      IF (ICHNG1.EQ.0) GO TO 580
C  PROCESS PREVIOUS CHANGE CARD
      IF (SERVRS(1).EQ.0) GO TO 560
C  CHANGE OF SERVERS
```

```
      I = 1
      M = 0
  551 DO 552 L=1,20
         IF (SERVRS(I).EQ.FACTYP(L)) GO TO 553
  552 CONTINUE
      GO TO 557
  553 J = FACQSX(L)
      IF (J.EQ.0) GO TO 557
      J = J-1
  554 I = I+1
      IFACNO = SERVRS(I)
      IF (IFACNO.EQ.0) GO TO 558
      IF (IFACNO.LT.0) GO TO 551
      IF (IFACNO.GT.NFACSM(L,1)) GO TO 557
      K1 = 11*(J+IFACNO-1)+1
      K2 = K1+1
C  CURRENT CONTENTS
      ICONT = ISTO(K1)
C  REMAINING CAPACITY
      IRCAP = ISTO(K2)
      I = I+1
      NEWCAP = SERVRS(I)
      IF (NEWCAP.LT.0) GO TO 557
      IF (NEWCAP.GE.ICONT) GO TO 555
C  MUST LOWER CAPACITY BELOW PRESENT CONTENTS USING STORAGE CHANGER
C  TRANSACTION IN GPSS
      ISTO(K2) = 0
      M = M+1
      IMAXBH(MH7(M,1)) = J+IFACNO
      IMAXBH(MH7(M+30,1)) = NEWCAP
      GO TO 554
C  MUST RAISE CAPACITY OR LOWER TO > OR = PRESENT CONTENTS. STORAGE
C  CHANGER XAC WILL LEAVE/ENTER TO RESTART DELAY CHAIN
  555 ISTO(K2) = NEWCAP-ICONT
      M = M+1
      IMAXBH(MH7(M,1)) = J+IFACNO
C  FIX ENTRY COUNT
      ISTO(K1+5) = ISTO(K1+5)-1
      GO TO 554
  557 WRITE (6,1101) TIME,SERVRS,I,M,L,J,IFACNO,K1,K2,
     * ICONT,IRCAP,NEWCAP
      CALL LOGIC (LS,JOBLS)
      GO TO 9 9 9 9 9
  558 DO 559 I=1,30
         SERVRS(I) = 0
  559 CONTINUE
      ISAVEH(NSCXH) = M
  560 CONTINUE
C
C  INSERT HERE ADDITIONAL CHANGE OPTIONS
C
C  READ NEXT CHANGE CARD
      READ (5,1002,END=585) ICARD
      NCARD = NCARD+1
      LINECT = LINECT+1
      IF (LINECT.LT.51) GO TO 579
      LINECT = 1
      WRITE (6,1005)
  579 WRITE (6,1004) NCARD,ICARD
C  ENTER HERE FIRST TIME THROUGH
  580 IF (ICARD(1).NE.ICHAN) GO TO 585
```

```
      ICHNG1 = 1
      CALL XCODE (BUFFER,80)
      WRITE (10,1002) ICARD
      BUFFER(1) = NAMECH
      BUFFER(2) = IAND(BUFFER(2),MASK2)+BLANK2
      CALL XCODE (BUFFER,84)
      READ (10,CH)
      IC = ISAVEH(CLKXH)
C  SET ADVANCE TIME
      ISAVEF(CHGXF) = 60*((TIME-(TIME/100)*40)-(IC-(IC/100)*40))
      GO TO 9 9 9 9 9
C  NO MORE CHANGES
  585 ISAVEF(CHGXF) = 1000000
      GO TO 9 9 9 9 9
C
C  LOWER STORAGE CAPACITY
C
  590 J = 11*(IVALUE(3)-1)+1
      NEWCAP = IVALUE(4)
      NURCAP = NEWCAP-ISTO(J)
      IF (NURCAP.GE.0) GO TO 592
      ISTO(J+1) = 0
      GO TO 9 9 9 9 9
  592 ISTO(J+1) = NURCAP
C  STORAGE LOWERING COMPLETE
      ISAVEH(SLCXH) = 1
      GO TO 9 9 9 9 9
C
   24         C    O    N    T    I    N    U    E
C
C...C O N C E S S I O N
C
C                   IVALUE(2) = CURRENT LOCATION - PH2
C                   IVALUE(3) = FLIGHT TABLE ROW - PH1
C                   IVALUE(4) = RANDOM NUMBER FOR CONC. AND LEAVE TIME
C                   IVALUE(5) = CLOCK - C1
C                   IVALUE(6) = SWITCH, =1 FOR LOBBY CONCESSION
C                                       =2 FOR CONCOURSE CONCESSION
C
      IF (NOCONC.EQ.0) GO TO 752
      NPTFM = IVALUE(2)
      IFLT = IVALUE(3)
      IGAT = IMAXBH(MH1(IFLT,9))
      I = 0
C  DETERMINE SECURITY FACILITY ASSIGNED TO GATE
      IF (IVALUE(6).EQ.2) I = IMAXBH(MH9(IGAT,4))
C  COUNT CONCESSIONS WITH SAME SECURITY
      L = INDEXF(15)+1
      M = INDEXF(15)+NOCONC
      IC = 0
      DO 751 J=L,M
         IF (IMAXBH(MH9(J,4)).EQ.I) IC = IC+1
  751 CONTINUE
      IF (IC.GT.0) GO TO 753
C  NO CONCESSION AVAILABLE
  752 CALL ASSIGN (5,0,PH)
      ISAVEH(TRVXH) = 0
      GO TO 9 9 9 9 9
C SELECT ONE CONCESSION RANDOMLY
  753 IRN = MOD(IVALUE(4),IC)+1
      IC = 0
```

```
      DO 754 J=L,M
         IF (IMAXBH(MH9(J,4)).EQ.1) IC = IC+1
         IF (IC.EQ.IRN) GO TO 755
  754 CONTINUE
  755 NPTTO = IMAXBH(MH9(J,3))
      ASSIGN 756 TO NEXT
      GO TO 950
  756 IC1 = IVALUE(5)
C  COMPUTE WHEN TO LEAVE CONCESSION
      ITIM = IMAXBH(MH1(IFLT,6))*60-IC1
      IF (IVALUE(6).EQ.1) ITIM = ITIM-LEAVEL-LEAVEV*IVALUE(4)/1000
      IF (IVALUE(6).EQ.2) ITIM = ITIM-LEAVEC-LEAVEV*IVALUE(4)/1000
      IF (ITIM.LT.0) ITIM = 0
      CALL ASSIGN (2,NPTTO,PH, 5,ITIM,PH, 7,J,PH, 11,15,PB)
      GO TO 9 9 9 9 9
C
   25         C    O    N    T    I    N    U    E
C
C
C...C O N C O U R S E
C
C                    IVALUE(2)=CURRENT LOCATION (PT. NO.=PH2)
C                    IVALUE(3)=GATE NUMBER--MH1(PH1,9)
C
      NPTFM=IVALUE(2)
      IV3=IVALUE(3)
      ISEC=IMAXBH(MH9(IV3,4))
      J=INDEXF(3)+ISEC
      NPTTO=IMAXBH(MH9(J,3))
      ASSIGN 920 TO NEXT
      GO TO 950
  920 CALL ASSIGN( 2,NPTTO,PH, 5,ISEC,PH )
      GO TO 99999
C
C
C...W A L K I N G   T I M E   C A L C U L A T I O N
C
C                   MH6 VALUES MAY BE MODIFIED IN ANY DESIRED MANNER HERE.
C
  950 IF(NPTOSW.EQ.1)GOTO 951
      IF(NPTFM.GT.0.AND.NPTTO.GT.0)GOTO 951
      NPTOSW=1
      WRITE(6,1032)NPTFM,NPTTO,IVALUE
  951 ISAVEH(TRVXH)=IMAXBH(MH6(NPTFM,NPTTO))
      ITEMPT=PVAL(PH,9)+ISAVEH(TRVXH)
      CALL ASSIGN( 9,ITEMPT,PH )
      GOTO NEXT, (309,313,326,338,
     *            516,521,526,531,536,
     *            691,719,
     *            727,756,
     *            813,856,861,874,920)
C
C
C...E R R O R   A B E N D
C
C                   IF ERROR COUNT EXCEEDS "ERRORS" (DEFAULT VALUE 50),
C                   PROGRAM WILL TERMINATE.
C
  999 WRITE(6,1999)
      CALL LOGIC(LS,JOBLS)
      GOTO 9 9 9 9 9
```

```
C
C
C
C    1    C    O    N    T    I    N    U    E
C    2    C    O    N    T    I    N    U    E
C    3    C    O    N    T    I    N    U    E
C    4    C    O    N    T    I    N    U    E
C    5    C    O    N    T    I    N    U    E
C    6    C    O    N    T    I    N    U    E
C    7    C    O    N    T    I    N    U    E
C    8    C    O    N    T    I    N    U    E
C    9    C    O    N    T    I    N    U    E
C   10    C    O    N    T    I    N    U    E
C   11    C    O    N    T    I    N    U    E
C   12    C    O    N    T    I    N    U    E
C   13    C    O    N    T    I    N    U    E
C   14    C    O    N    T    I    N    U    E
C   15    C    O    N    T    I    N    U    E
C   16    C    O    N    T    I    N    U    E
C   17    C    O    N    T    I    N    U    E
C   18    C    O    N    T    I    N    U    E
C   19    C    O    N    T    I    N    U    E
C   20    C    O    N    T    I    N    U    E
C   21    C    O    N    T    I    N    U    E
C   22    C    O    N    T    I    N    U    E
C   23    C    O    N    T    I    N    U    E
C   24    C    O    N    T    I    N    U    E
C   25    C    O    N    T    I    N    U    E
C
C
C
99999     R    E    T    U    R    N
C
C
C
 1000 FORMAT(' ERROR IN FLIGHT INPUT DATA CARD.')
 1001 FORMAT(/,'   WARNING.   NO CHECKIN FACILITY DEFINED FOR AIRLINE CO
     *E',I3,'.   FACILITY OF AIRLINE CODE',I3,' USED.   RESULTS UNPREDIC
     *TABLE.')
 1002 FORMAT(20A4)
 1003 FORMAT(' ERROR IN GEOMETRY CARD.   INVALID FACILITY TYPE IN CARD S
     *EQUENCE',I4,'.')
 1004 FORMAT(2X,I4,3X,20A4)
 1005 FORMAT(1H1,///,15X,'I N P U T   D A T A',//)
 1006 FORMAT(//,10X,'E N D   O F   I N P U T   D A T A',//)
 1007 FORMAT(/,'   WARNING.   PROBLEM IN "GROUND TRANSPORT MODE" LOGIC.
     *  PASSENGER ASSIGNED TO BUS.   CHECK GRTRANSP DATA.')
 1008 FORMAT(/,'   WARNING.   ATTEMPT TO EXIT TO BLOCK NUMBER',I5,' VIA
     *"EXIT".   RESULTS UNPREDICTABLE.   CHECK FUNCTION',I3,'.')
 1009 FORMAT(/,'   WARNING.   ATTEMPT TO EXIT TO DEPLANING CURB FROM FAC
     *ILITY TYPE ',A4,'.   RESULTS UNPREDICTABLE.')
 1010 FORMAT(/,'   WARNING.   PASSENGER ATTEMPTED TO GO TO IMMIGRATION.
     *  NO FACILITIES DEFINED.   RESULTS UNPREDICTABLE.')
 1011 FORMAT(/,'   WARNING.   NO IMMIGRATION FACILITY SPECIFIED FOR GATE
     *',I3,'.  ',I3,' CHOSEN.')
 1012 FORMAT(/,'   WARNING.   ATTEMPT TO EXIT TO DEPLANING CURB FROM ',A
     *4,'.   RESULTS UNPREDICTABLE.   CHECK FUNCTION',I3,'.')
 1013 FORMAT(/,'   WARNING.   NO SECURITY FACILITY DEFINED FOR GATE',I3,
     *'.   SECURITY FACNO 1 ASSIGNED.   CHECK GATE INPUT CARD FOR IPARAM
     *(2).',/,'   THIS MESSAGE WILL NOT REPEAT.')
 1014 FORMAT(/,'   WARNING.   GATE',I4,' NOT DEFINED.   CHECK DATA FOR D
```

```
     *EPARTING FLIGHT',I5,'.   GATE',I4,' USED.',/,'    RESULTS UNPREDICT02134000
     *ABLE.',/,'   THIS MESSAGE WILL NOT REPEAT.')                      02135000
 1015 FORMAT(/,'   WARNING.   INVALID CALL TO FORTM "PARKING".   PH2=',I02136000
     *4,', PH4=',I5,', PB7=',I2,', PB6=',I2,'.   RESULTS UNPREDICTABLE.'02137000
     *)                                                                 02138000
 1016 FORMAT(/,'   ERROR.   VEHICLE XAC',I5,' UNABLE TO MATCH WITH PAX A02139000
     *T DEPLANING CURB.   CHECK USER CHAIN "ERROR" FOR THIS XAC.',      02140000
     */,'   RESULTS UNPREDICTABLE.')                                    02141000
 1017 FORMAT(/,'   ERROR.   PAX XAC WITH GROUND TRANSPORT MODE',I3,' ENT02142000
     *ERED BLOCK DPLCO.   CHECK USER CHAIN "ERROR" FOR XAC NO',I5,'.',  02143000
     */,'   RESULTS UNPREDICTABLE.')                                    02144000
 1018 FORMAT(/,'   WARNING.   NO FACILITY DEFINED FOR CAR RENTAL AGENCY,02145000
     * CODE',I3,'.   FACILITY FOR AGENCY CODE',I3,' USED.   RESULTS UNPR02146000
     *EDICTABLE.')                                                      02147000
 1019 FORMAT(/,'   WARNING.   NO CAR RENTAL FACILITIES DEFINED.   RESULT02148000
     *S UNPREDICTABLE.   THIS MESSAGE WILL NOT REPEAT.')                02149000
 1020 FORMAT(///,'   WARNING.   NO FACILITIES HAVE BEEN DEFINED FOR THE 02150000
     *FOLLOWING CLASSES:')                                              02151000
 1021 FORMAT(11X,A8)                                                    02152000
 1022 FORMAT(/,'   EXECUTION CONTINUES.')                               02153000
 1023 FORMAT(/,'   WARNING.   ADDITION OF DEPARTING FLIGHT, MH1 ROW NO',02154000
     *I4,' TO TRANSFER FLIGHT TABLE MH5 WOULD HAVE CREATED OVERFLOW COND02155000
     *ITION.',/,'   FLIGHT NOT ADDED.   EXECUTION CONTINUES.')          02156000
 1024 FORMAT(///)                                                       02157000
 1025 FORMAT('   WARNING.   POINTX AND POINTY BOTH 0 FOR POINT',I4,'.') 02158000
 1026 FORMAT(11X,'TICKETS&CHECKIN')                                     02159000
 1027 FORMAT(/,'   ERROR.   NO TICKETS&CHECKIN FACILITY DEFINED FOR AIRL02160000
     *INE CODE',I3,'.   FACILITY OF AIRLINE CODE',I3,' USED.')          02161000
 1028 FORMAT(/,'   ERROR.   NO "TICKETS&CHECKIN" FACILITIES DEFINED FOR 02162000
     *ENPLANING PASSENGERS.   RUN TERMINATED.')                         02163000
 1029 FORMAT(/,'   ERROR.   NO EXPRESS CHECKIN FACILITY DEFINED FOR AIRL02164000
     *INE CODE',I3,'.   FULL SERVICE FACILITY OF AIRLINE CODE',I3,' USED02165000
     *.')                                                               02166000
 1030 FORMAT(11X,'IMMIGRATION')                                         02167000
 1031 FORMAT(/,' *** ERROR IN INPUT DATA.   DOUBLE DEFINITION OF ',A8,' 02168000
     *NUMBER ',I3,'.   RUN TERMINATED (SEE FORTM, STATEMENT NO. 269).') 02169000
 1032 FORMAT(/,' *** NON-POSITIVE POINT NUMBER IN WALKING TIME CALC.',/'02170000
     *   NPTFM = ',I4,'; NPTTO = ',I4,/,'   IVALUE:',6I8,/,'   RESULTS N02171000
     *OT PREDICTABLE.   THIS MESSAGE WILL NOT REPEAT.')                 02172000
 1033 FORMAT('   FROM SECTION 18 - TRANSFER FLIGHT.   IVALUE(2)= ',I3,  02172100
     *'   IVALUE(3)= ',I3)                                              02172200
 1050 FORMAT(1H1,///,(2X,64A2))                                         02173000
 1051 FORMAT(      /,38X,'B O A R D I N G   G A T E   F A C I L I T Y   02174000
     * R E P O R T',///)                                                02175000
 1052 FORMAT(      /,36X,'E X P R E S S   C H E C K I N   F A C I L I T02176000
     * Y   R E P O R T',///)                                            02177000
 1053 FORMAT(      /,42X,'S E C U R I T Y   F A C I L I T Y   R E P O R02178000
     * T',///)                                                          02179000
 1055 FORMAT(      /,43X,'C U S T O M S   F A C I L I T Y   R E P O R T02180000
     *',///)                                                            02181000
 1061 FORMAT(      /,34X,'C A R   R E N T A L   A G E N C Y   F A C I L02182000
     * I T Y   R E P O R T',///)                                        02183000
 1063 FORMAT(      /,39X,'I M M I G R A T I O N   F A C I L I T Y   R E02184000
     * P O R T',///)                                                    02185000
 1064 FORMAT(      /,32X,'T I C K E T I N G   &   C H E C K I N   F A C02186000
     * I L I T Y   R E P O R T',///)                                    02187000
 1070 FORMAT(      /,35X,'5   M I N U T E   S N A P S H O T S',//)      02188000
 1071 FORMAT(1X,'CLOCK   PAX ENTERING   PAX LEAVING   UAL EXPRESS   UAL 02189000
     *TICKETING   BRANIFF   PAX ENTERING CONCOURSE      PAX LEAVING CONC02190000
     *OURSE')                                                           02190100
 1072 FORMAT(1X,'TIME      TERMINAL       TERMINAL      PAX FLOW      PA02191000
```

```
     *X FLOW       PAX FLOW',I3,6I4,1X,7I4,//)
 1073 FORMAT(1X,I5,I11,I15,I13,I15,I14,1X,7I4,1X,7I4)
 1074 FORMAT(/,25X,'5   M I N U T E   S N A P S H O T S   O F   C O N G
     *E S T I O N   A T   P O I N T S',//)
 1082 FORMAT(5X,'CLOCK  POINT')
1083  FORMAT(5X,'CLOCK TIME',10X,'UNTD COTL FRNT     PRK1 SECB SECC SECD
     1')
 1076 FORMAT(1X,'TIME          1    2    3    4    5    6    7    8    9
     * 10   11   12   13   14   15   16   17   18   19   20   21   22
     *23   24',//)
 1077 FORMAT(1X,I4,4X,24I5)
 1095 FORMAT(9X,24I5)
 1097 FORMAT(100I5)
 1092 FORMAT(17X,'F A C I L I T Y   U T I L I Z A T I O N',30X,'Q U E U
     *E   S T A T I S T I C S',//)
 1094 FORMAT(4X,'FACILITY    NO. OF    TOTAL NO.    MAX. NO. OF    AVG. NO.
     * OF    AVG. TIME      TOTAL QUEUE   MAX. QUEUE   AVG. QUEUE    AVG.
     *TIME')
 1096 FORMAT(4X,' NUMBER    AGENTS    OF PATRONS   AGENTS BUSY   AGENTS B
     *USY    PER PATRON       ENTRIES          SIZE          SIZE       IN QU
     *EUE',//)
 1075 FORMAT(7X,I3,7X,I2,7X,I4,10X,I2,11X,F5.2,8X,I2,':',I2,11X,I4,10X,
     *        I3,9X,F5.2,7X,I2,':',I2,/)
 1078 FORMAT(///,10X,'(ALL TIMES IN MINUTES:SECONDS)')
 1079 FORMAT(/,10X,'** INDICATES UNDEFINED (UNLIMITED) NO. OF AGENTS.')
 1080 FORMAT(/,'   WARNING.   TITLE CARDS LIMITED TO 5.  ABOVE TITLE CAR
     *D WILL NOT BE PRINTED.',/)
 1081 FORMAT(8X,64A1)
 1101 FORMAT(/,'   ERROR.   CHANGE CARD INCORRECT.   RUN TERMINATED.'/
     * I10/(10I10))
 1999 FORMAT(///,'      *** E R R O R   E N D ***   -   PROGRAM TERMINA
     *TING DUE TO ERROR COUNT EXCEEDING "ERRORS".')
 1087 FORMAT(25I5)
 1088 FORMAT(5X,I4,7X,F7.2,5X,F7.2,6X,F7.2,5X,F7.2)
 1089 FORMAT(16X,F7.2,5X,F7.2,6X,F7.2,5X,F7.2)
 1090 FORMAT(2X,I4,2X,F7.2,2X,F7.2,2X,F7.2,2X,F7.2,2X,F7.2,2X,F7.2,2X,
     XF7.2,2X,F7.2)
C
      END
```

# APPENDIX B-4

# ALSIM DOCUMENTATION - SUBROUTINES

## FORTRAN Subroutine CLINK

## Assembler Subroutines CLINK1 and CLINK2

PURPOSE:

These subroutines perform a linking operation, allowing GPSS HELPA blocks to operate as HELPC blocks. Both block types are used to call FORTRAN subroutines, however, when HELPC is executed, the called subprogram obtains routine access to GPSS entities and Standard Numerical Attributes contained in the B-through G-operands. HELPA blocks normally only provide one way communication between the GPSS main program and the FORTRAN subroutines.

The HELPC procedure requires GPSS to construct the entity address argument list in a specific order each time a HELPC block is utilized, then GPSS passes control to the FORTRAN subprogram. This argument list is identical for all HELPC calls. Using this linking procedure, the subroutines CLINK, CLINK1 and CLINK2 store addresses of these arguments within the called FORTRAN subprogram and eliminate the need for constructing the argument list repeatedly. Any HELPA call executed after use of these subroutines provides the required access to argument values for two-way communication between the GPSS main program and the FORTRAN subroutine.

USAGE:

A FORTRAN subprogram using the capabilities of these subroutines must contain a secondary entry point. The name of this entry point must be used as a member of the data set for

link editing. This FORTRAN subprogram must be kept resident in main memory during similation through use of the LOAD feature of GPSS and loaded under the name of the secondary entry point. The entry point name must be used as an alias to the name of the subroutine. For example, the subroutine LINKC has an entry point FORTM. The link edited member name would appear as LINKC (FORTM).

The linking subroutines described here are used to establish the required argument list addresses of the FORTRAN subroutine by a two step process. The GPSS program calls the FORTRAN subroutine CLINK using a HELPC block. This is coded as in the following example;

HELPC CLINK, 1.

Immediately following this block is a HELPA call to the entry point of the FORTRAN subprogram requiring access to GPSS entities and SNA values. Using the previous member names, an example of this HELPA call is the following;

HELPA FORTM, 1, 1.

The B-operand of the HELPC call may take on any value, but must be identical to the C-operand of the HELPA block. The purpose of using these values is to designate a location in the GPSS fullword save value storage area to temporarily store the argument list addresses.

The FORTRAN subprogram CLINK contains an argument list constructed according to the format specified by GPSS. Addresses of the variables used as arguments will be stored within the

FORTRAN subprogram LINKC and be available for reference when the subprogram is executed through the entry point FORTM.

The following example illustrates the FORTRAN statements required to utilize the linking subroutines. The FORTRAN subprogram is named LINKC, as before, and contains the entry point FORTM as shown.

```
SUBROUTINE LINKC (IVALUE, ISAVEF, ISAVEH, IFAC, ISTO, FSTO,
*IQUE, FQUE, ILOG, ITAB, FTAB, IUSE, IUSEF, FUSE, IMAX, IMAXB,
*IMAXH, IMAXBH, FSAVEL, IMAXL, FMAXBL)
REAL *8 FQUE, FUSE, FTAB
INTEGER *2 ISAVEH, ILOG, IUSE, IMAXBH.
DIMENSION IVALUE(6), ISAVEF(2), ISAVEH(2), IFAC(2), ISTO(2), FSTO(2),
*IQUE(2), FQUE(2), ILOG(2), ITAB(2), FTAB(2), IUSE(2), IUSEF(2),
*FUSE(2), IMAX(2), IMAXB(2), IMAXH(2), IMAXBH(2), FSAVEL(2),
*IMAXL(2) FMAXBL(2)
         .
         .
         .
RETURN
         .
         .
         .
ENTRY FORTM (IVALUE)
         .
         .
         .
CALL CLINK2
```

Note that the LINKC argument list contains the B through G-operands in IVALUE and the GPSS entity reference words, but this subroutine is not called directly by the GPSS program.

Instead, another FORTRAN subroutine, CLINK, is called by a HELPC block in the GPSS program. The CLINK argument list is identical to that of LINKC. Subroutine CLINK will call the assembler program CLINK1 to store the CLINK argument list in a fullword savevalue area of GPSS and then return to GPSS.

After CLINK returns to GPSS, an immediate HELPA call to FORTM results in a call to CLINK2. The assembler subroutine CLINK2 subsequently calls LINKC. The argument addresses in the GPSS fullword savevalue area will be transferred to the LINKC subprogram and stored, making them accessible every time a call to FORTM is performed. CLINK2 returns to FORTM which, in turn, returns to GPSS. The simulation then operates with HELPA blocks calling the secondary entry point of the FORTRAN subprogram and performing the same functions as HELPC blocks.

RESTRICTIONS:

These subroutines were written to conform with code internal to the DAG05 module of the IBM GPSS-V program product. Attempts to use it with other versions of GPSS may yield unpredictable results.

Subroutine LINKC violates the constraint that a FORTRAN subroutine may not call itself or any other subroutine which subsequently calls it. A FORTRAN compiler more sophisticated than the IBM G-1, release 2.0 version may prohibit this operation.

Fullword savevalues used to store the argument list of CLINK should not contain information for retention prior to calling CLINK. The contents of this area are not retained by

any of these subroutines.

PROGRAM LOGIC:

CLINK

The subprogram CLINK contains an argument list built by GPSS which references variables in a specified order and which must be stored in a GPSS savevalue area. Subroutine CLINK calls assembler subprogram CLINK1 to perform this storage operation. After CLINK1 returns to CLINK, this subroutine returns to the GPSS main program.

PROGRAM LOGIC:

CLINK1

This program saves all registers except 13 and designates 12 as the base register. The GPSS save area address is obtained from the CLINK save area by displacing register 13 by 4 bytes. The contents of GPSS registers 1 and 10 are obtained at locations 24 and 60 bytes into the GPSS save area and loaded into registers 10 and 11, respectively. Register 10 then contains the address of a 25 word table established by GPSS. The starting address of GPSS control words is found at a location 24 bytes within this table. This address is loaded into register 10. A displacement of 1044 bytes from register 10 contents provides the address of the starting location of GPSS fullword savevalues. This address is next loaded into register 10 for later use in locating an area to store the CLINK argument list.

The GPSS argument list address was obtained from GPSS register 1 and is now contained in register 11. Contents stored

at this address, which are the address of the B operand of the GPSS HELPC call, are loaded into register 2. The value, N, of the B operand is subsequently loaded into register 2 and the contents are shifted left by 2 bits. This value is added to the address in register 10. The resulting address used to store the argument list then begins at a location N words into the fullword savevalue storage area.

Addresses of the CLINK argument variables, starting with IVALUE, are loaded into registers 0 through 9. These addresses are stored in locations beginning at the address indicated in register 10. Subsequent load and store instructions place the argument addresses in 21 contiguous fullword savevalue locations.

The program executes a RETURN macro instruction to restore all registers except 13 from the CLINK save area, then branches back to CLINK.

PROGRAM LOGIC:

CLINK2

Assembler subroutine CLINK2 is called by the FORTRAN subprogram LINKC from a location following FORTM, the secondary entry point. Subroutine CLINK2 subsequently calls the FORTRAN subprogram LINKC, which contains the entry point and the call to CLINK2.

CLINK2 executes the SAVE macro to store the contents of all registers except 13 and declares 12 as the base register. Register 11 is loaded with the address of the FORTRAN subprogram

LINKC, which contains the entry point and the call to CLINK2.

CLINK2 executes the SAVE macro to store the contents of all registers except 13 and declares 12 as the base register. Register 11 is loaded with the address of the FORTRAN subprogram save area from register 13. The starting address of an 18 fullword save area, SAVEA, is loaded into register 13. The address of SAVEA is stored 8 bytes into the FORTRAN subprogram save area and the FORTRAN subprogram save area starting address is stored 4 bytes after the address SAVEA. The address of the FORTRAN subprogram save area is also stored in the first word of the 19 fullword storage area FORTSAVE. The contents of the FORTRAN save area are also stored in the remaining 18 words of FORTSAVE.

The starting address of the GPSS program save area is then obtained from the location 4 bytes beyond the start of the FORTRAN save area and loaded into register 11. The contents of GPSS register 1, the GPSS argument list starting address, are obtained and loaded into register 1 from the GPSS save area. The address of the first argument, the B-operand, is obtained from the location specified by register 1 and is loaded into register 1. The value of the C-operand is then loaded into register 1 from the location 4 bytes beyond the address of the first argument. Register 1 contents are then shifted left by 2 bytes.

The GPSS program savevalue area is again accessed and the contents of GPSS register 10 are loaded into register 10. CLINK2 then obtains the starting address of fullword savevalues

in the manner identical to CLINK1 and places it in register 10. The CLINK argument list address in the fullword savevalue area is obtained by adding registers 1 and 10 and placing their sum in register 1.

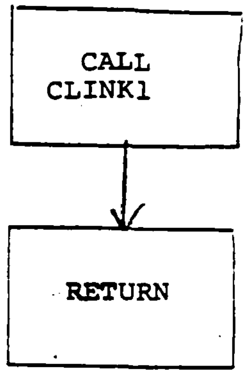
Because CLINK2, an assembler program, calls the FORTRAN subprogram LINKC, a branch to IBCOM is performed to provide a traceback capability if the program terminates when the FORTRAN subprogram is operating. Upon return from IBCOM, the program branches to the FORTRAN subprogram, LINKC.

The fullword savevalue storage address used for storing the CLINK address list is contained in register 1 at this time. The argument variables are identical to those of LINKC. When LINKC is called, the SAVE macro is executed and this address is saved with other register contents in the save area, SAVEA. The FORTRAN compiler also obtains the argument list address stored in the GPSS fullword savevalue area from register 1 and then stores the addresses of the arguments in contiguous storage locations within the FORTRAN subprogram LINKC. After performing this storage function, control is passed back to CLINK2. The fullword savevalue area used to store the argument list is no longer required for that purpose and is made zero through a series of load multiple and store multiple instructions.

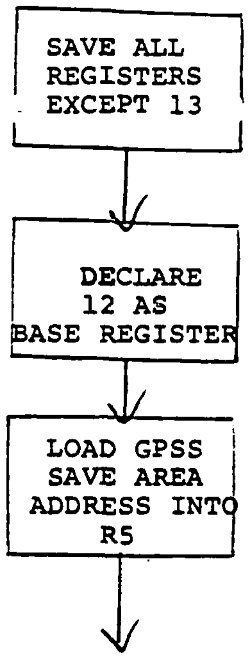
The address of the FORTRAN subprogram save area, contained in the first word of FORTSAVE, is placed in register 11. The contents of FORTSAVE, established by CLINK2 to store the FORTRAN subroutine save area, are placed in the FORTRAN subroutine

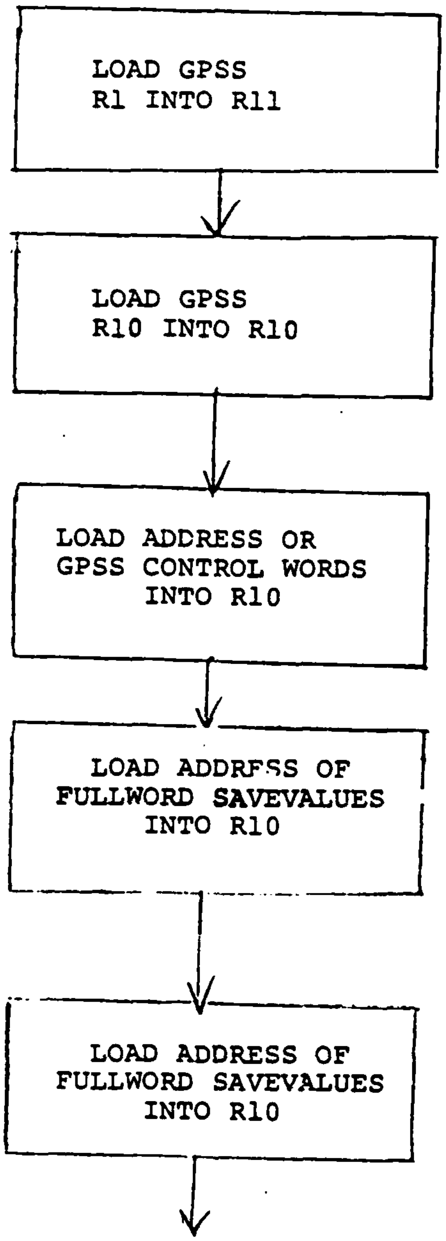
save area. Register 13 is loaded with the address of the FORTRAN subprogram save area from the second word of the CLINK2 save area, SAVEA. The program executes a RETURN macro to restore registers from the FORTRAN subprogram save area and returns control to the FORTRAN subprogram.

CLINK




CLINK1




LOAD GPSS
R1 INTO R11
LOAD GPSS
R10 INTO R10
LOAD ADDRESS OR
GPSS CONTROL WORDS
INTO R10
LOAD ADDRESS OF
FULLWORD SAVEVALUES
INTO R10
LOAD ADDRESS OF
FULLWORD SAVEVALUES
INTO R10

LOAD B ARGUMENT
(IVALUE(1)) INTO
R2

↓

SHIFT R2 LEFT 2 BITS

↓

ADD R10 AND R2
TO SET SAVEVALUE
LOCATION, RESULT
IN R10

↓

LOAD FIRST 10
WORDS OF CLINK
ARGUMENT LIST
IN R0 TO R9

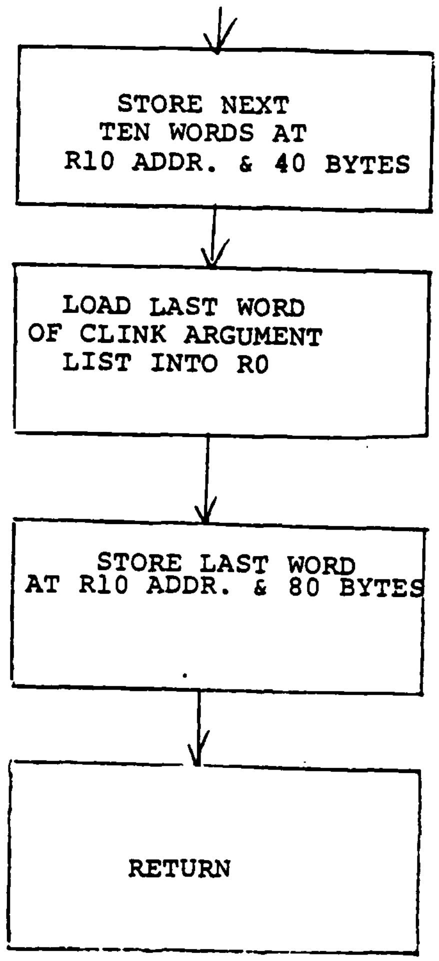
↓

STORE FIRST 10
WORDS AT LOCATION
BEGINNING AT
R10 ADDRESS

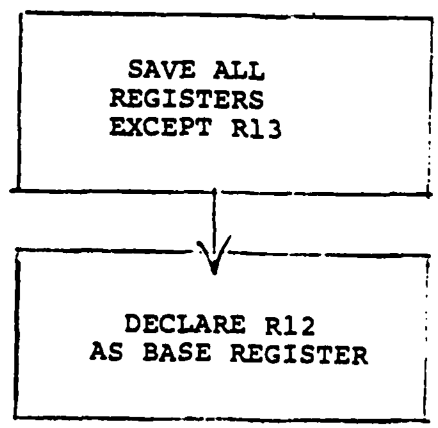
↓

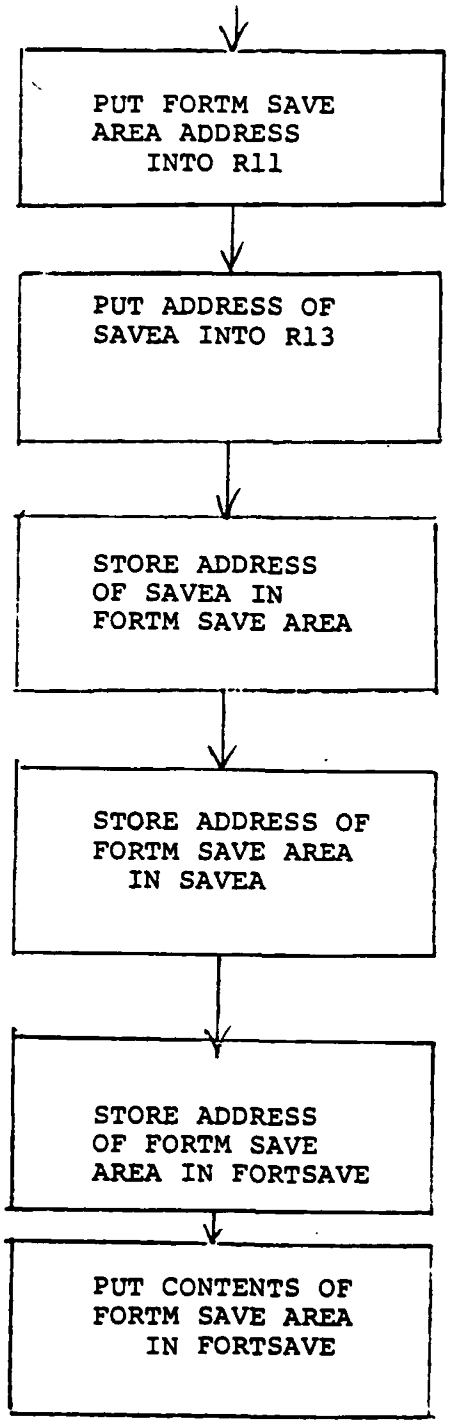
LOAD NEXT 10
WORDS INTO
R0 TO R9

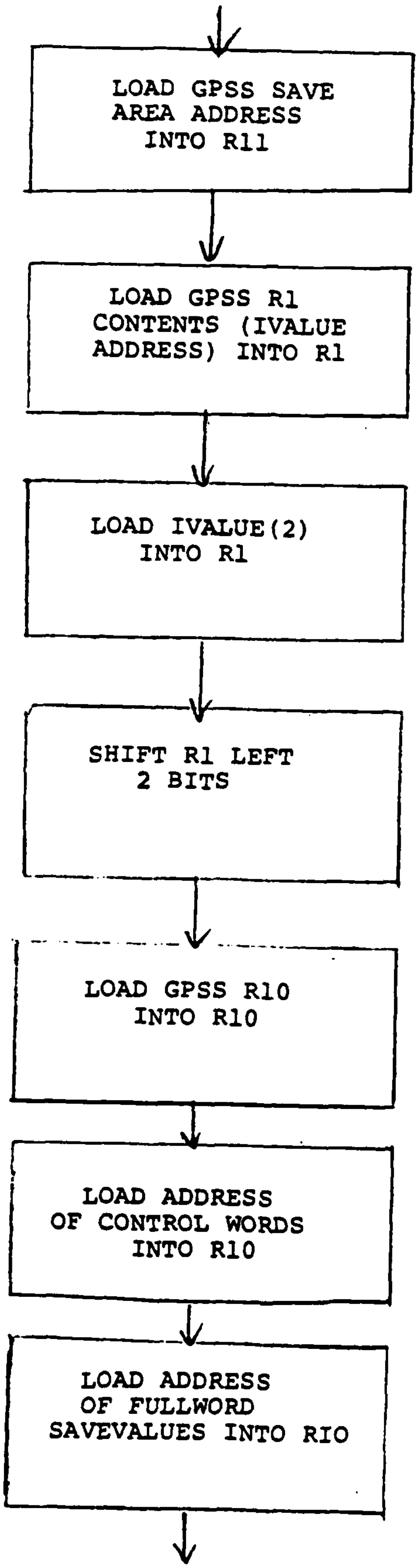
↓

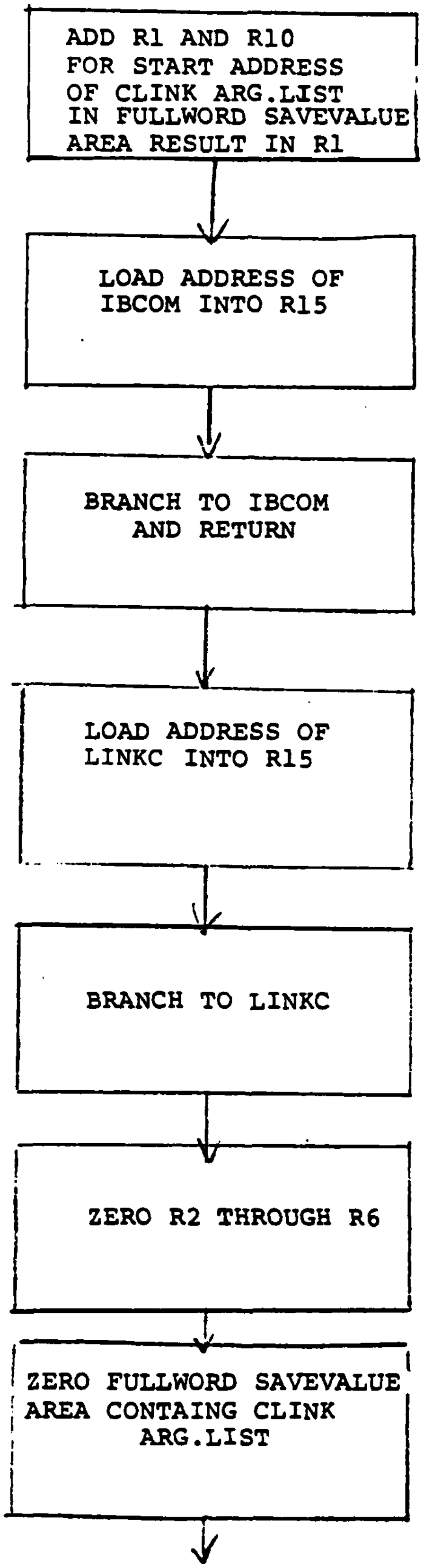

STORE NEXT
TEN WORDS AT
R10 ADDR. & 40 BYTES
LOAD LAST WORD
OF CLINK ARGUMENT
LIST INTO R0
STORE LAST WORD
AT R10 ADDR. & 80 BYTES
RETURN

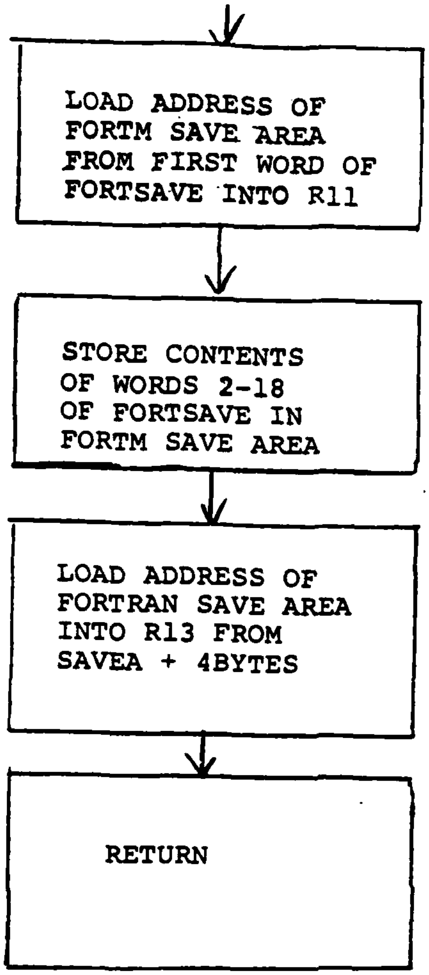
CLINK 2


SAVE ALL
REGISTERS
EXCEPT R13
DECLARE R12
AS BASE REGISTER


PUT FORTM SAVE AREA ADDRESS INTO R11
PUT ADDRESS OF SAVEA INTO R13
STORE ADDRESS OF SAVEA IN FORTM SAVE AREA
STORE ADDRESS OF FORTM SAVE AREA IN SAVEA
STORE ADDRESS OF FORTM SAVE AREA IN FORTSAVE
PUT CONTENTS OF FORTM SAVE AREA IN FORTSAVE


LOAD GPSS SAVE AREA ADDRESS INTO R11
LOAD GPSS R1 CONTENTS (IVALUE ADDRESS) INTO R1
LOAD IVALUE(2) INTO R1
SHIFT R1 LEFT 2 BITS
LOAD GPSS R10 INTO R10
LOAD ADDRESS OF CONTROL WORDS INTO R10
LOAD ADDRESS OF FULLWORD SAVEVALUES INTO RIO


ADD R1 AND R10 FOR START ADDRESS OF CLINK ARG.LIST IN FULLWORD SAVEVALUE AREA RESULT IN R1
LOAD ADDRESS OF IBCOM INTO R15
BRANCH TO IBCOM AND RETURN
LOAD ADDRESS OF LINKC INTO R15
BRANCH TO LINKC
ZERO R2 THROUGH R6
ZERO FULLWORD SAVEVALUE AREA CONTAING CLINK ARG.LIST


LOAD ADDRESS OF FORTM SAVE AREA FROM FIRST WORD OF FORTSAVE INTO R11
STORE CONTENTS OF WORDS 2-18 OF FORTSAVE IN FORTM SAVE AREA
LOAD ADDRESS OF FORTRAN SAVE AREA INTO R13 FROM SAVEA + 4BYTES
RETURN

```
MEMBER NAME  CLINK
      SUBROUTINE CLINK(IVALUE,ISAVEF,ISAVEH,IFAC,ISTO,FSTO,IQUE,
     *FQUE,ILOG,ITAB,FTAB,IUSE,IUSEF,FUSE,IMAX,IMAXB,IMAXH,IMAXBH,FSAVEL
     *,IMAXL,FMAXBL)
      INTEGER*2 ISAVEH,ILOG,IUSE,IMAXBH
      REAL*8 FQUE,FUSE,FTAB
      DIMENSION IVALUE(6),ISAVEF(2),ISAVEH(2),IFAC(2),ISTO(2),FSTO(2),
     *IQUE(2),FQUE(2),ILOG(2),ITAB(2),FTAB(2),IUSE(2),IUSEF(2),FUSE(2),
     *IMAX(2),IMAXB(2),IMAXH(2),IMAXBH(2),FSAVEL(2),IMAXL(2),FMAXBL(2)
      CALL CLINK1
      RETURN
      END

MEMBER NAME  CLINK1
CLINK1   START 0
         SAVE  (14,12),,*
         BALR  12,0
         USING *,12
         L     5,4(13)
         L     11,24(5)          GPSS R1
         L     10,60(5)          GPSS R10
         L     10,24(10)
         L     10,1044(10)       XF AREA
         L     2,0(11)           IVALUE
         L     2,0(2)            IVALUE(1)
         SLA   2,2
         AR    10,2
         LM    0,9,0(11)
         STM   0,9,0(10)
         LM    0,9,40(11)
         STM   0,9,40(10)
         L     0,80(11)
         ST    0,80(10)
         RETURN (14,12),T
         END
```

```
MEMBER NAME  CLINK2
CLINK2   START 0
         EXTRN LINKC
         SAVE  (14,12),,*
         BALR  12,0
         USING *,12
         LR    11,13
         LA    13,SAVEA          CLINK2 SAVE
         ST    13,8(11)          TO FORT SAVE (CHAIN)
         ST    11,4(13)          FM FORT SAVE (CHAIN)
         ST    11,FORTSAVE
         LM    2,10,0(11)
         STM   2,10,FORTSAVE+4
         LM    2,10,36(11)
         STM   2,10,FORTSAVE+40
         L     11,4(11)
         L     1,24(11)          GPSS R1
         L     1,0(1)            IVALUE
         L     1,4(1)            IVALUE(2)
         SLA   1,2
         L     10,60(11)         GPSS R10
         L     10,24(10)
         L     10,1044(10)       XF AREA
         AR    1,10              ARG LIST ADDR
         L     15,=V(IBCOM#)
         BAL   14,64(15)
         L     15,ADLINKC
```

```
         BALR  14,15
         SR    2,2
         LR    3,2
         LR    4,2
         LR    5,2
         LR    6,2
         STM   2,6,0(1)
         STM   2,6,20(1)
         STM   2,6,40(1)
         STM   2,6,60(1)
         ST    2,80(1)
         L     11,FORTSAVE
         LM    2,10,FORTSAVE+4
         STM   2,10,0(11)
         LM    2,10,FORTSAVE+40
         STM   2,10,36(11)
         L     13,SAVEA+4
         RETURN (14,12),T
FORTSAVE DS    19F
SAVEA    DS    18F
ADLINKC  DC    A(LINKC)
         END
```

## ASSEMBLER SUBROUTINE MNLINK

PURPOSE:

This subroutine provides a method for passing numerical values of GPSS-V mnemonics used in the Airport Landside simulation model to supporting FORTRAN subroutines during program execution. This feature allows development of FORTRAN subprograms independently without reference to absolute values assigned by the operation of GPSS-V. Data for output under FORTRAN format control is also passed from GPSS-V through mnemonic linking. Types of information transmitted are: savevalues, GPSS entity identifiers, numbers of columns of halfword matrices and GPSS program locations.

USAGE:

An explicitly numbered GPSS-V list function containing mnemonics to be passed must be established after the last mnemonic referenced. A FORTRAN CALL statement to MNLINK must contain the absolute function number as the first argument. The remaining arguments are positionally identified with GPSS mnemonics appearing on the list as Y values. It is desirable, though not necessary, to use similar or identical arguments and Y list names. The lists may be expanded indefinitely.

The list function is placed near the end of a GPSS-V program, as illustrated in the following example:

```
.
.
.
1 FUNCTION PH1, L4
, CMH01/, CMH02/, CML02/, CLKXH
START 1,,,1
END
```

A HELPA or HELPC block transfers control to the FORTRAN subprogram. Generally, the mnemonic link is activated by the first FORTRAN call of the simulation. Contained in the FORTRAN instruction set is the call to MNLINK, as shown:

```
CALL MNLINK(1, CMH01, CMH04, CML02, CLKXH).
```

The numerical value 1 of the first argument is in agreement with the GPSS-V identification number of the list function. After the return from MNLINK, FORTRAN argument names appearing in the CAL statement have the absolute values of GPSS-V names appearing in corresponding positions of the function.

RESTRICTIONS:

1. All member names of the argument list must be FORTRAN fullword integers.

2. Mnemonics appearing in the list function must be unique names, i.e. each mnenomic must be used for only one purpose.

3. The FORTRAN calling program must be kept loaded with the GPSS program during the simulation or MNLINK must be called each time the FORTRAN subroutine is loaded.

4. The subroutine was written to conform with code internal to the DAG05 module of IBM GPSS-V. Attempts to use this assembly program with other versions of GPSS-V may yield unpredictable results.

PROGRAM LOGIC:

The MNLINK subroutine executes the SAVE macro to retain contents of all registers except 13 and specifies 12 as the base register. The FORTRAN save area address is obtained from register 13. The second word of the area contains the address of the GPSS-V save area and is loaded into register 10. From the GPSS save area, contents of GPSS registers 2 and 3 are placed in the corresponding program registers. Contents of GPSS registers 10 and 11 are also loaded into MNLINK registers 10 and 11. This is performed to locate a GPSS 25 word table and to allow entry into the GPSS subroutine UNFLOT. In addition, register 14 contents are made 4096 greater than those of register 2 as required for entry into GPSS routines.

The 25 word table established by GPSS, with a starting address in register 10 contains the starting address of UNFLOT. A displacement of 80 bytes into the list points to the starting address of the UNFLOT routine. This address is placed in register 7 and subsequently in the fullword storage defined as UNFLOT.

The address of GPSS control words is contained in the table at a displacement of 24 bytes. These control words provide the starting address of GPSS entities. A displacement of 1052 bytes in the control word area provides the starting address of functions. Register 10 is loaded with this address.

The number of the function is the first entry of the FORTRAN argument list and is located at the address contained in register 1. The function number stored at this address is loaded into register 6.

Because each function occupies 32 bytes, apart from Y values, a left shift of the function number in register 6 by 5 bits allows indexing of the function addresses. After the left shift, the required list function address is located by adding registers 6 and 10. The number of points or mnemonics is located 12 bytes into this function area. The value at this address is placed in register 6. The starting address of Y values is contained in the first byte and is loaded into register 10.

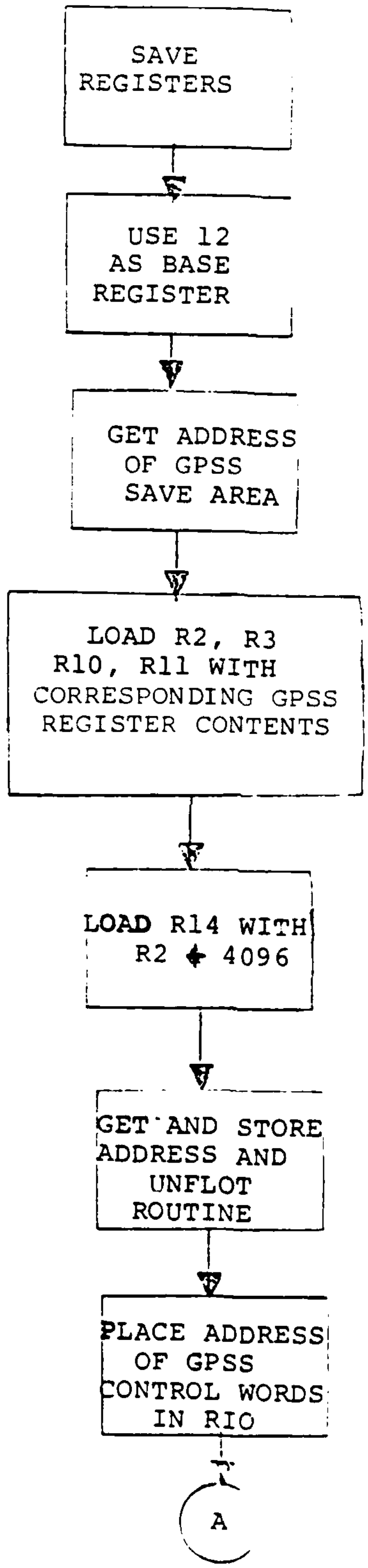
A value of four is stored in register 7 to increment registers 1 and 10 through the respective argument list and Y value addresses. Register 1 is pointed to the second word of the argument list. Register 2 is established as a floating point register and the contents are zeroed.

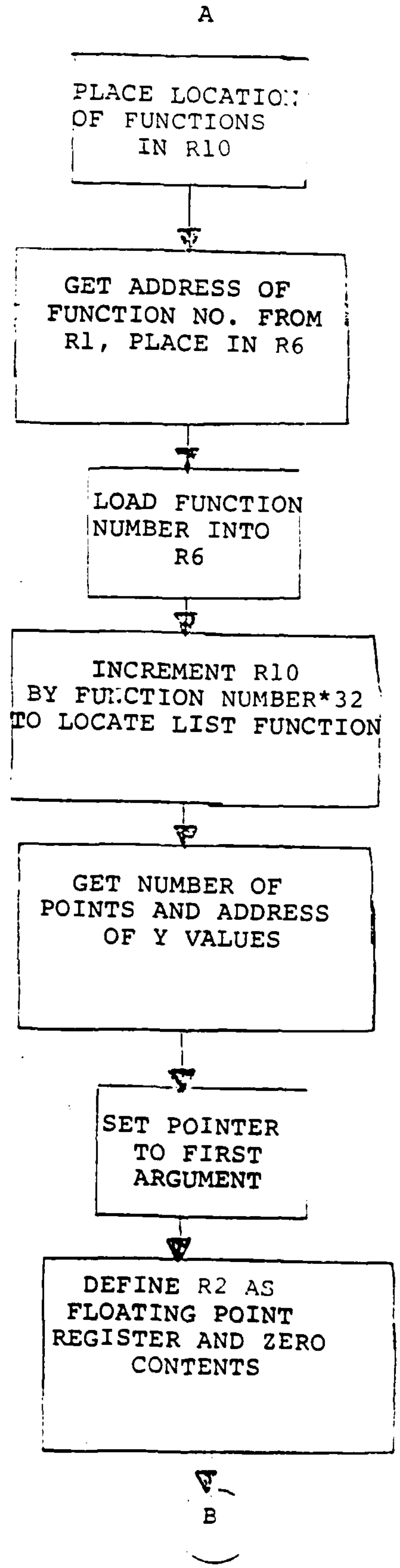
A loop to process the word list begins at the address NEXTPT. Register 4 is first pointed to the address of the second word of the argument list and register 2 is loaded with the value of the first Y point. The GPSS subroutine UNFLOT is called to convert the floating point Y value of the list function contained in register 2 to an integer.

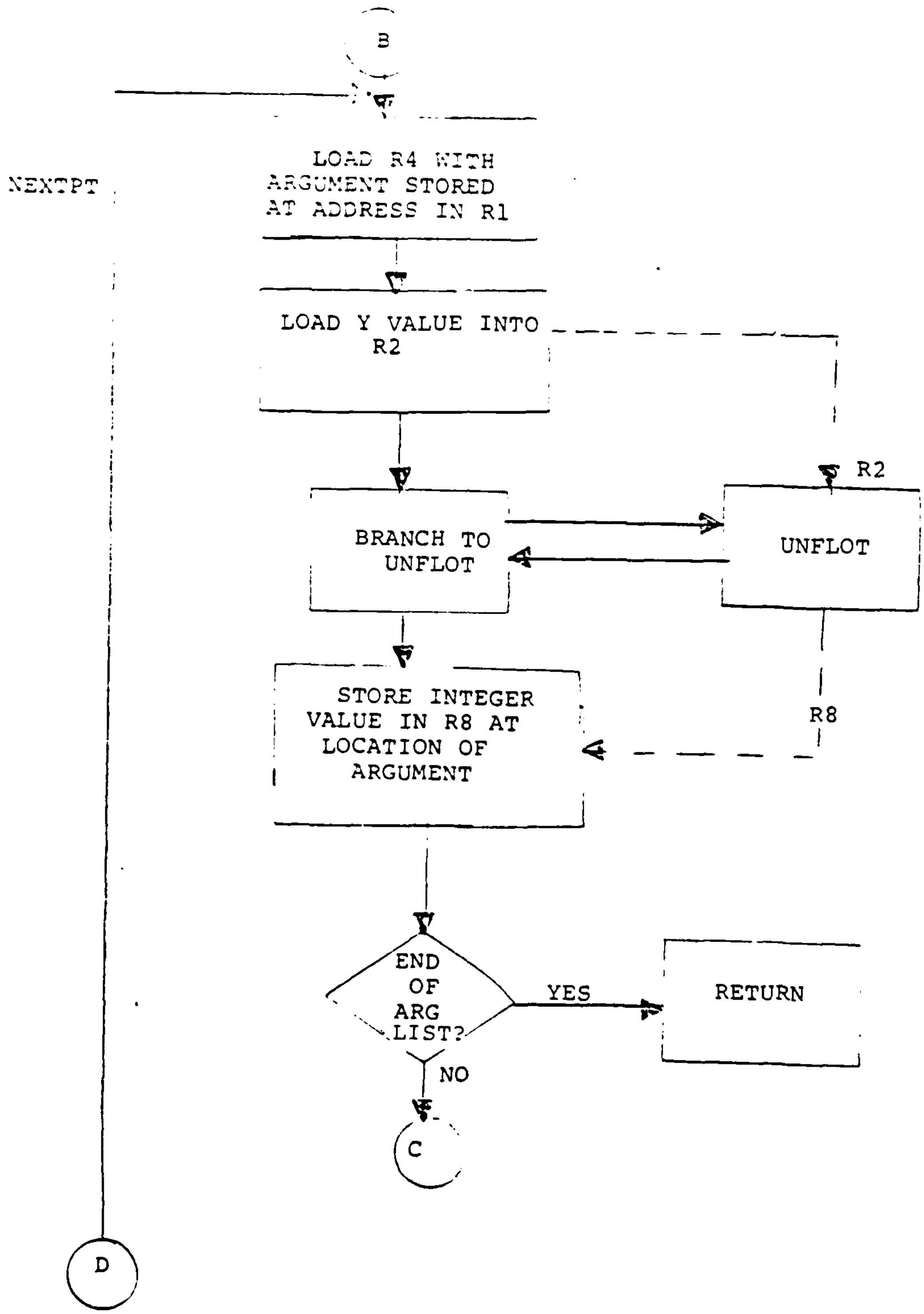
The integer portion of the value returned by UNFLOT is contained in register 8 and the fraction portion in number 9. The value in register 9, being zero, is ignored. The register 8 result is stored in the argument list location specified in register 4. Thus the absolute values of the entities contained in the link function are stored in the MNLINK argument list locations for later reference.

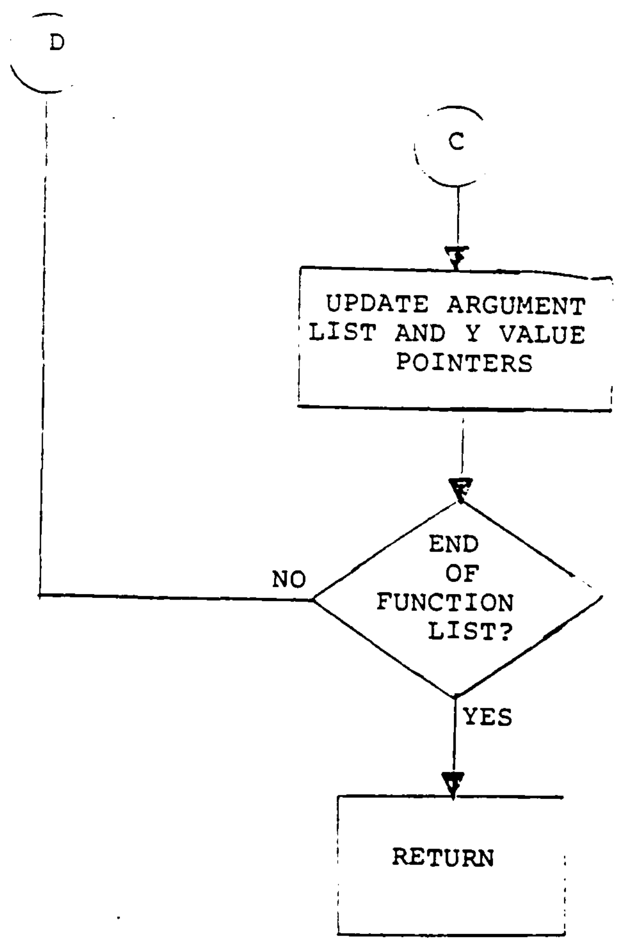
The subroutine tests for the end of the argument list. If another mnemonic is to be linked, registers 1 and 10 are incremented by 4 bytes. The list function length is decremented by one in register 6 and compared to zero. If register 6 is greater than zero, the program returns to NEXTPT where register 4 is pointed to the next address in the agrument list, and register 2 is loaded with the value of the next Y point. If register 4 has a negative sign bite, indicating the argument list end, the program restors the general registers and returns to the FORTRAN calling location.

MNLINK


SAVE REGISTERS
USE 12 AS BASE REGISTER
GET ADDRESS OF GPSS SAVE AREA
LOAD R2, R3 R10, R11 WITH CORRESPONDING GPSS REGISTER CONTENTS
LOAD R14 WITH R2 + 4096
GET AND STORE ADDRESS AND UNFLOT ROUTINE
PLACE ADDRESS OF GPSS CONTROL WORDS IN R10
A


A
PLACE LOCATION OF FUNCTIONS IN R10
GET ADDRESS OF FUNCTION NO. FROM R1, PLACE IN R6
LOAD FUNCTION NUMBER INTO R6
INCREMENT R10 BY FUNCTION NUMBER*32 TO LOCATE LIST FUNCTION
GET NUMBER OF POINTS AND ADDRESS OF Y VALUES
SET POINTER TO FIRST ARGUMENT
DEFINE R2 AS FLOATING POINT REGISTER AND ZERO CONTENTS
B


B
NEXTPT
LOAD R4 WITH ARGUMENT STORED AT ADDRESS IN R1
LOAD Y VALUE INTO R2
R2
BRANCH TO UNFLOT
UNFLOT
STORE INTEGER VALUE IN R8 AT LOCATION OF ARGUMENT
R8
END OF ARG LIST?
YES
RETURN
NO
C
D


D
C
UPDATE ARGUMENT LIST AND Y VALUE POINTERS
END OF FUNCTION LIST?
NO
YES
RETURN

```
MNLINK   START 0
         SAVE  (14,12),,*
         BALR  12,0
         USING *,12
         SR    0,0
         L     10,4(13)
         LM    2,3,28(10)
         LM    10,11,60(10)
         LR    14,2
         A     14,=F'4096'
         L     7,80(10)          ADDR OF UNFLOT ROUTINE
         ST    7,UNFLOT
         L     10,24(10)
         L     10,1052(10)
         L     6,0(1)            FN NO ADDR (FORT)
         L     6,0(6)            FN NO
         SLA   6,5(0)
         AR    10,6              R10 POINTS TO FN AREA
         LH    6,12(10)          NO OF POINTS
         L     10,0(10)          ADDR OF Y-VALUES
         LA    7,4
         AR    1,7
         SDR   2,2
NEXTPT   L     4,0(1)            POINT TO NEXT FORT CALL ARG ADDR
         LE    2,0(10)           FN VALUE
         L     15,UNFLOT
         BALR  5,15
         ST    8,0(4)
         CR    4,0               TEST END OF ARG LIST
         BNH   RETURN
         AR    1,7               UPDATE ARG LIST POINTER
         AR    10,7              UPDATE FUNCTION POINTER
         BCT   6,NEXTPT
RETURN   RETURN (14,12)
UNFLOT   DS    1F
         END
```

## Assembler Subroutine XCODE

PURPOSE:

This subroutine permits FORTRAN programs to perform in-core read and write operations. XCODE provides the capability for rereading input data, and is similar in this respect to the READRE routine available at many 360/370 installations. However, because it operates on arrays in main storage instead of on I/O buffers, flexibility may be attained in performing reformatting operations. A particular example of this application to NAMELIST data is used in the Airport Landside Simulation Model.

USAGE:

Subroutine XCODE requires the designation of a data set reference number and an array to act as a buffer area. The data set must not be identified by a DD card. The buffer area array must be large enough to accomodate all read or write operations involving the designated data set.

XCODE must be called prior to each read or write operation involving the designated data set. The calling statement has the following form:

CALL XCODE (array name, length of I/O operation in bytes).

The following example illustrates a use of XCODE. An 80-column data card is read under an A format into the array ICARD. The characters are subsequently written into the array, BUFFER, and reread from this array under a NAMELIST format.

Character data is used to test for data card type and to place the NAMELIST special form characters at the beginning and end of the record. Two card types, PARM and AIRLINE, are shown in this example. For each of these, a call is made to XCODE with the arguments BUFFER and buffer size 80. After the return to FORTRAN, a WRITE statement places the ICARD data into the BUFFER array.

A subsequent call is made to XCODE with BUFFER and 84 as the array and buffer size arguments, respectively. The ensuing READ statement uses the 21 words of BUFFER to perform a NAMELIST read operation. Device 10 is not specified by a DD statement.

```
      DIMENSION ICARD (20), BUFFER (21)
      DATA NAMEPA, NAMEAL, NAMEND/' &PA', ' &AL', '&END'/
      DATA IPARM, IARLIN, IBLANK/'PARM', 'AIRL', '    '/
      NAMELIST/PA/BOARDT, GREET, WWGATE, GRGATE, CRBCHK
      NAMELIST/AL/LINES, EPCURB, BUSTOP, EXPCHK
      BUFFER (21) = NAMEND
  101 READ (5, 1000) ICARD
 1000 FORMAT (20 A4)
      IF (ICARD (1). EQ. IPARM) GO TO 1
      IF (ICARD (1). EQ. IARLIN) GO TO 2
                .
                .
                .
 1    ICARD (1) = NAMEPA
      CALL XCODE (BUFFER, 80)
```

```
      .
      WRITE (10, 1000) ICARD
      CALL XCODE (BUFFER, 84)
      READ (10, PA)
      .
      .
      .
2     ICARD (1) = NAMEAL
      ICARD (2) = BLANK
      CALL XCODE (BUFFER, 80)
      WRITE (10, 1000) ICARD
      CALL XCODE (BUFFER, 84)
      READ (10, AL)
```

Input data cards for this example are shown below. Card identifiers do not require the NAMELIST special form, but only the literal symbols PARM and AIRLINE. Data items are treated as keyword parameters using variable names identified by NAMELIST statements. A blank separates card identifiers and other symbols. Columns 1 through 80 are available for card identification plus data.

```
PARM WWGATE = 19, GRGATE = 12, GREET = 43
AIRLINES LINES = 1, EPCURB = 3, EXPCHK = 70
```

PROGRAM LOGIC:

XCODE

The subroutine declares 15 as the base register and saves registers 14 through 3 in the FORTRAN calling program save area. The addresses of the two calling arguments are obtained from the argument list address contained register 1 and

loaded into registers 2 and 3 respectively. The value of the second argument, the buffer size, is obtained from the address contained in register 3 and placed in that register. Register 2 contains the starting address of the array BUFFER. The contents of registers 2 and 3 are stored in the 2 fullword storage area BUFFADDR.

The program then places the entry point address XCODE2 in register 1, the address CLOAD in register 3, and branches to CLOAD. Register 3 is declared the base register and the address of IBCOM is placed in register 15 to satisfy base register requirements in IBCOM. The program places a hexidecimal 50 in the location 74 bytes within IBCOM thereby changing the IBCOM instruction;

```
L 1,VFIOCS
```

to become,

```
ST 1,VFIOCS
```

The program executes the second instruction and stores XCODE2 at the address VFIOCS. The LOAD instruction is restored with a second MOVE IMMEDIATE instruction. XCODE proceeds back to the branch instruction where it restores registers 14 to 3 from the save area, zeroes out register 15 and returns to the FORTRAN subprogram.

The next FORTRAN WRITE or READ instruction is processed by IBCOM. At some point during IBCOM execution, a branching to the address contained in VFIOCS results in a branch to XCODE2 because of the previous substitution.

At XCODE2 contents of register 4 are saved at SAVEAREA.

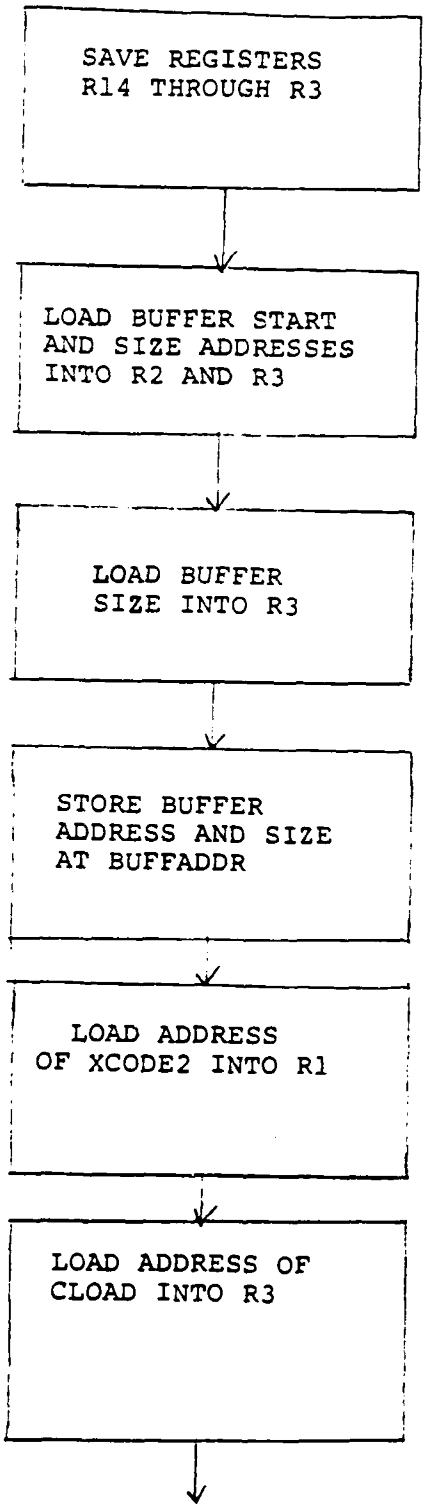
The address of XCODE2 is loaded into register 4 from register 1. Register 4 is declared the base register.

Register 0 contains an address constant from IBCOM. This value is loaded into register 1. Contents of storage one byte beyond the location indicated by register 1 are tested by a test under mask instruction. If the pass through the XCODE2 section arises from a FORTRAN WRITE statement, branching to location OUTPUT is executed. At this location, register 2 is loaded with the starting address of BUFFER bytes to be written onto. The first byte of BUFFER is blanked by a hex '40'. Subsequent bytes are blanked by decrementing register 3 twice and executing the MVC instruction at DMOVE. This operation is performed on the array BUFFER, up to a limit of register 3 contents plus one times. Register 3 is then incremented by two to again contain the number of BUFFER bytes specified for writing. At RETURN, the program restores register 4 and places the IBCOM arguments in register 1. A branch to 6 bytes beyond register 1 contents returns control to IBCOM, where writing of input data into BUFFER is completed.

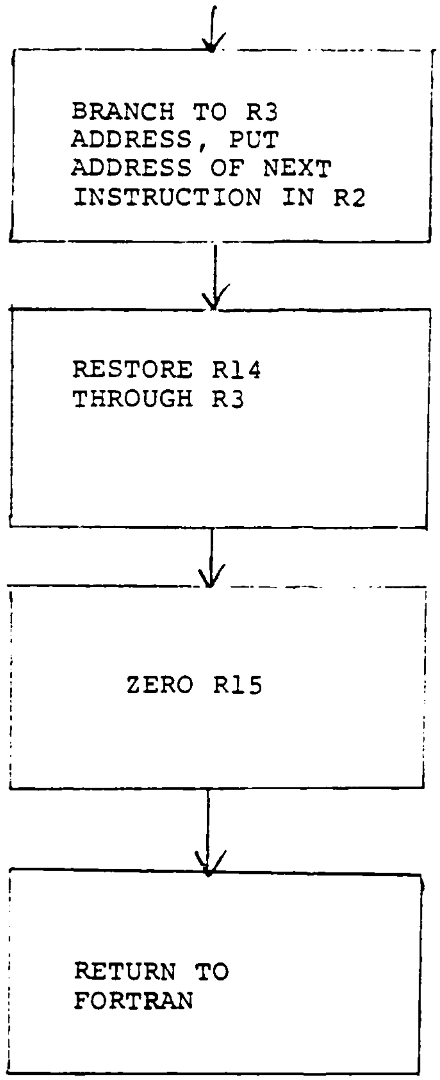
A FORTRAN READ statement also causes branching to XCODE2 from IBCOM. However, the program does not branch to OUTPUT. Instead, the program loads the address of FIOCS into register 1 and the address of CLOAD into register 3. XCODE branches to CLOAD and declares register 3 as the base register. The address of FIOCS is restored to IBCOM by performing the instruction at CLOAD and subsequent instructions.

Following this replacement, the program branches back to place the two fullwords in BUFFADDR into registers 2 and 3. The program branches to RETURN and subsequently returns control to IBCOM for execution of the in-core read under namelist format control.

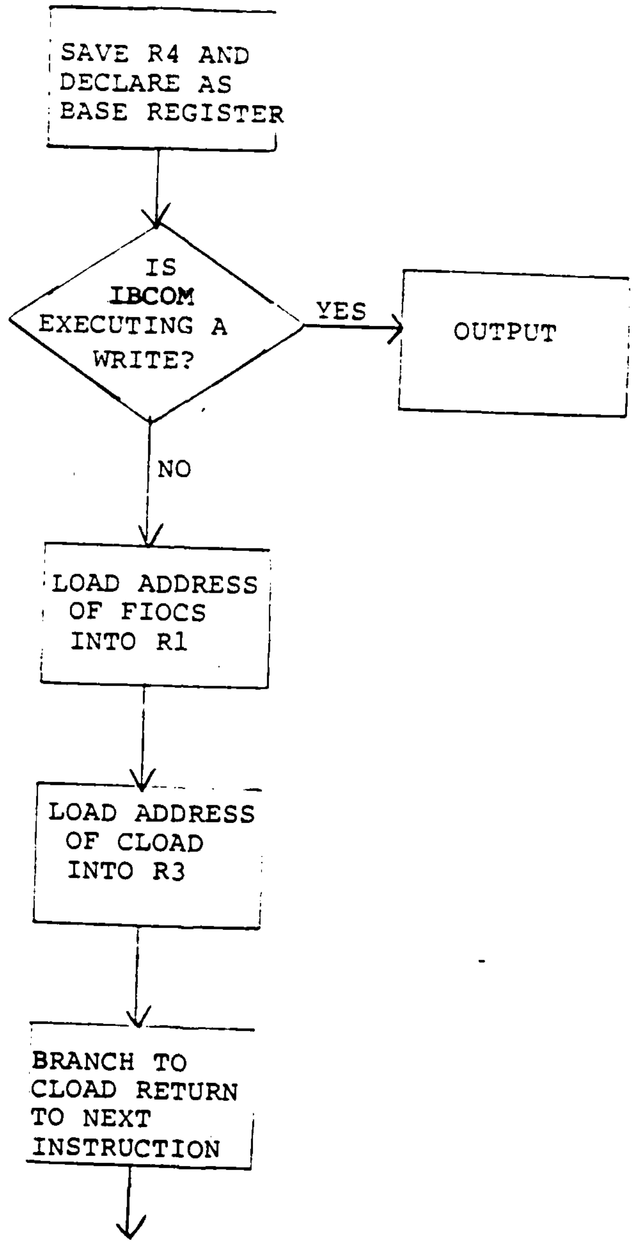
XCODE

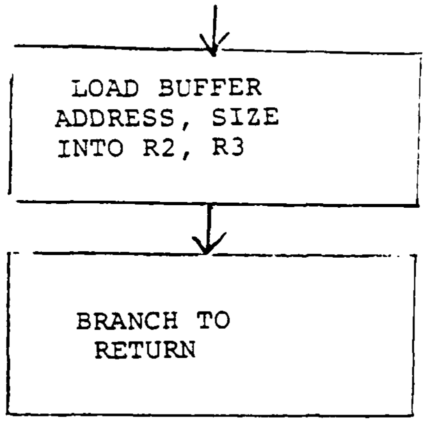


SAVE REGISTERS R14 THROUGH R3
LOAD BUFFER START AND SIZE ADDRESSES INTO R2 AND R3
LOAD BUFFER SIZE INTO R3
STORE BUFFER ADDRESS AND SIZE AT BUFFADDR
LOAD ADDRESS OF XCODE2 INTO R1
LOAD ADDRESS OF CLOAD INTO R3

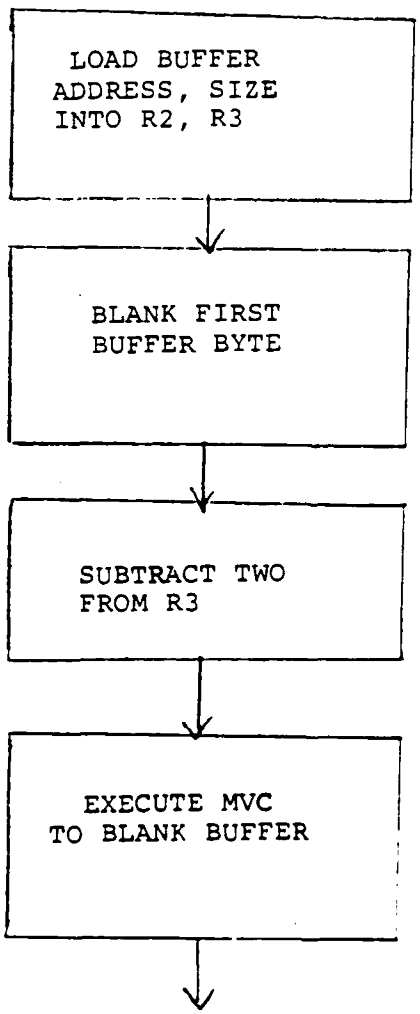

BRANCH TO R3 ADDRESS, PUT ADDRESS OF NEXT INSTRUCTION IN R2
RESTORE R14 THROUGH R3
ZERO R15
RETURN TO FORTRAN

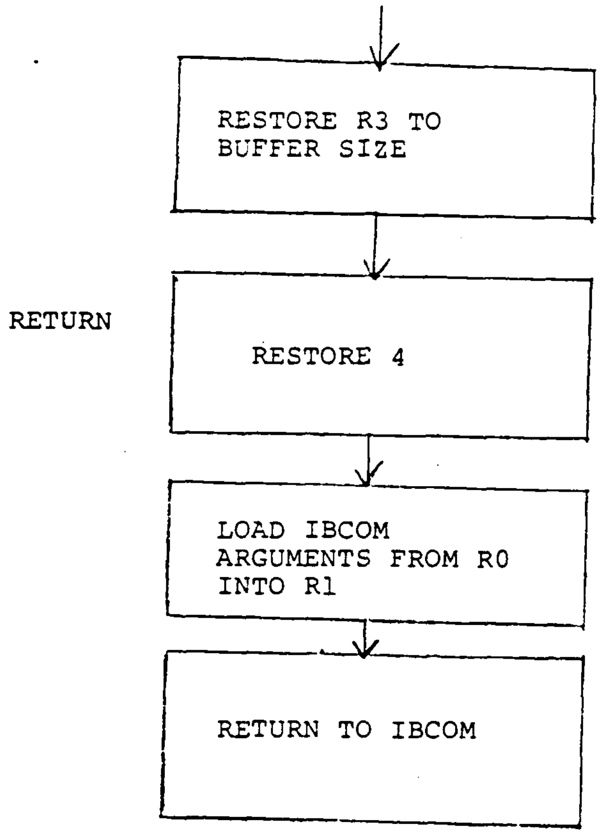
XCODE2

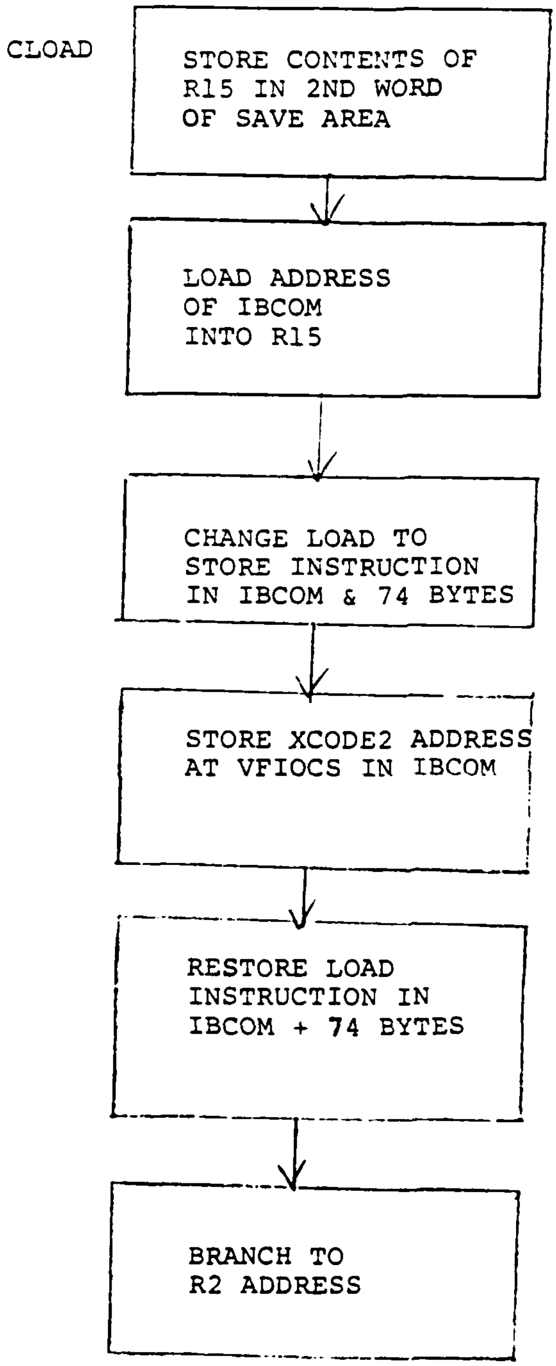

SAVE R4 AND DECLARE AS BASE REGISTER
IS IBCOM EXECUTING A WRITE?
YES
OUTPUT
NO
LOAD ADDRESS OF FIOCS INTO R1
LOAD ADDRESS OF CLOAD INTO R3
BRANCH TO CLOAD RETURN TO NEXT INSTRUCTION


LOAD BUFFER ADDRESS, SIZE INTO R2, R3
BRANCH TO RETURN

OUTPUT


LOAD BUFFER ADDRESS, SIZE INTO R2, R3
BLANK FIRST BUFFER BYTE
SUBTRACT TWO FROM R3
EXECUTE MVC TO BLANK BUFFER


RESTORE R3 TO BUFFER SIZE
RETURN
RESTORE 4
LOAD IBCOM ARGUMENTS FROM R0 INTO R1
RETURN TO IBCOM


CLOAD
STORE CONTENTS OF R15 IN 2ND WORD OF SAVE AREA
LOAD ADDRESS OF IBCOM INTO R15
CHANGE LOAD TO STORE INSTRUCTION IN IBCOM & 74 BYTES
STORE XCODE2 ADDRESS AT VFIOCS IN IBCOM
RESTORE LOAD INSTRUCTION IN IBCOM + 74 BYTES
BRANCH TO R2 ADDRESS

```
MEMBER NAME  XCODE
XCODE#   START 0
         ENTRY XCODE
         EXTRN IBCOM#
         EXTRN FIOCS#
         USING *,15
XCODE    B     *+14
         DC    XL4'07000000'
         DC    CL6'XCODE '
         STM   14,3,12(13)
         LM    2,3,0(1)
         L     3,0(3)
         STM   2,3,BUFFADDR
         LA    1,XCODE2
         LA    3,CLOAD
         BALR  2,3
         LM    14,3,12(13)
         SR    15,15
         BR    14
         DROP  15
         USING *,1
XCODE2   ST    4,SAVEAREA
         LR    4,1
         USING XCODE2,4
         DROP  1
         LR    1,0
         TM    1(1),X'0F'
         BO    OUTPUT
         L     1,ADRFIOCS
         LA    3,CLOAD
         BALR  2,3
         LM    2,3,BUFFADDR
         B     RETURN
```

```
OUTPUT   LM    2,3,BUFFADDR
         MVI   0(2),X'40'
         BCTR  3,0
         BCTR  3,0
         EX    3,DMOVE
         LA    3,2(3)
RETURN   L     4,SAVEAREA
         LR    1,0
         DROP  4
         B     6(1)
DMOVE    MVC   1(0,2),0(2)
         USING *,3
CLOAD    ST    15,SAVEAREA+4
         L     15,ADRIBCOM
         MVI   74(15),X'80'
         EX    0,74(15)
         MVI   74(15),X'58'
         L     15,SAVEAREA+4
         BR    2
BUFFADDR DS    2F
SAVEAREA DS    2F
ADRIBCOM DC    A(IBCOM#)
ADRFIOCS DC    A(FIOCS#)
          END
```

## FORTRAN FUNCTION - MHBASE, MXBASE, MLBASE

PURPOSE:

These functions provide the base addresses of GPSS-V halfword, fullword and floating point matrices used in the FORTRAN section of the airport landside simulation model. Computed base addresses are used by FORTRAN statement functions to compute addresses of GPSS-V matrix elements for data insertion and extraction. The use of this function or a similar algorithm for referencing GPSS matrices by a FORTRAN program is necessitated by the incompatibility of GPSS internal storage with the FORTRAN array structure. This subprogram and associated FORTRAN statement functions permit addressing of program matrix elements by row and column symbols.

USAGE:

This subroutine is link edited with the primary name MHBASE and aliases MXBASE and MLBASE. The only calls to this function occur on the first HELPC call from GPSS. Each matrix requires a separate call with the following syntax:

```
                     MHBASE (IMAXH,
FORTRAN variable =   MXBASE (IMAX, Matrix No., No. of cols.)
                     MLBASE (IMAXL,
```

Variables IMAXH, IMAX and IMAXL are arguments passed from GPSS when a HELPC call is made. The matrix number is specified explicitly for each call to the function. The number of columns is initially specified in the GPSS program by an SYN statement. The GPSS symbol used in the statement is identified with the FORTRAN variable representing the number of columns in the matrix by the mnemonic link function. The value assigned to the GPSS symbol must agree with the number of columns specified in the GPSS matrix definition statement.

As an example, a matrix to be utilized in the simulation is halfword matrix number 2, consisting of 15 rows and 7 columns. The number of columns is identified with the symbol CMHO2 by the following GPSS SYN statement:

```
CMHO2 SYN 7 NO. OF COL - MH2
```

The GPSS matrix definition statement establishing halfword matrix 2 is the following:

```
2 MATRIX MH , 15, 7
```

The mnemonic link function must cor' 'n a reference to CMHO2;

1 FUNCTION PH1, L 20 MNEMONIC LINK FUNCTION, CMHO1,/CMHO2,/ CMHO3,/....

A positional correspondence between CMHO2 and the variable ICNHO2 is established by the FORTRAN call to MNLINK

CALL MNLINK (1, ICNHO1, ICNHO2, ICNHO3, .......)

The call to MHBASE to establish the base address of MH2 is illustrated by the following FORTRAN statement:

MHO2B = MHBASE (IMAXH, 2, ICNHO2)

This base address is used by the following FORTRAN statement function to calculate the address of the element in the IR row, IC column of halfword matrix 2.

MH2 (IR, IC) = MH02B + ICNHO2*IR + IC

RESTRICTIONS:

Standard mnemonics and indexing constants used in coding HELPC routines are used in this function. This subprogram requires that versions of GPSS-V used for simulation contain these conventions.

PROGRAM LOGIC:

The subprogram name MHBASE is used to designate this function. The calling argument IMAXH is dimensioned 1 as are IMAX and IMAXL. The base address MHBASE of halfword matrix N is calculated by the following expression:

MHBASE = IMAXH (6*N-5)/2 - ICN - 1

The variable ICN represents the number of columns in halfword matrix N.

At entries MXBASE and MLBASE, base addresses of fullword and floating point matrices respectively are calculated using expressions of the same form. After each base address calculation, the program returns to the calling FORTRAN subprogram.

FUNCTION MHBASE (IMAXH,N,ICN)

COMPUTE
MHBASE

RETURN

ENTRY MXBASE (IMAX,N,ICN)

COMPUTE
MXBASE

RETURN

ENTRY MLBASE (IMAXL,N,ICH)

COMPUTE
MLBASE

RETURN

```
MEMBER NAME  MHBASE
      FUNCTION MHBASE(IMAXH,N,ICN)
      DIMENSION IMAX(1),IMAXH(1),IMAXL(1)
C
C     MUST BE ASSIGNED ALIASES OF MXBASE AND MLBASE.
C
      MHBASE=IMAXH(6*N-5)/2-ICN-1
      RETURN
C
      ENTRY MXBASE(IMAX,N,ICN)
      MXBASE=IMAX(6*N-5)/4-ICN-1
      RETURN
C
      ENTRY MLBASE(IMAXL,N,ICN)
      MLBASE=IMAXL(6*N-5)/4-ICN-1
      RETURN
      END
```

## ASSEMBLER SUBROUTINE ASSIGN/LOGIC/PVAL/FPVAL

PURPOSE:

This subroutine allows a FORTRAN subroutine called by a GPSS-V HELPC or HELPA block to perform the function of the GPSS ASSIGN block. Furthermore, this subroutine executes the set and reset functions of the GPSS LOGIC block and obtains parameter values directly from the currently active GPSS transaction. This subroutine is called by the FORTRAN subroutine.

USAGE:

This subroutine must be link edited with the name ASSIGN and aliases LOGIC, PVAL and FPVAL. The FORTRAN subroutine ARGERR must be a member of SYS1. FORTLIB or in a user library concatenated with SYS1. FORTLIB at link edit time.

The calling FORTRAN subprogram must contain the following statements:

```
INTEGER * 2 LR, LS, PB, PF, PH, PL
INTEGER PVAL
DATA LR, LS, PB, PF, PH, PL/'LR', 'LS', 'PB', 'PF', 'PH',
         'PL'.
```

During the simulation run, the active GPSS transaction calls the FORTRAN subprogram through a HELPA or HELPC block. Parameters of that transaction are assigned values by using the following call statements in the FORTRAN subprogram;

```
CALL ASSIGN (parameter number, FORTRAN variable or
              constant, parameter type).
```

Multiple assignments and mixed parameters are valid. This is exhibited in the following example;

```
CALL ASSIGN (1, 10, PH, 3, XRAY, PL, 1, IVAL, PF, 5,
              3, PB).
```

When a logic switch requires a set or reset condition, the FORTRAN program executes the following subroutine call;

```
CALL LOGIC (logic set (LS)/logic reset (LR), switch
            number)
```

Multiple sets and resets and mixed types are valid, as

shown in the following call statement:

CALL LOGIC (LS, 1, LS, 3, LR, 4).

When a parameter value of the active transaction is required, the FORTRAN program uses the functions PVAL or FPVAL. The statements used to obtain this value for integer parameters are:

FORTRAN variable = PVAL (type, parameter number),
For floating point parameters, the value is obtained
by using FORTRAN variable = FPVAL (PL, parameter number)

The valid PVAL function parameter types are PF, PH, or PB. Floating point parameters, PL, are evaluated by FPVAL. Only one parameter may be referenced in a statement. The following example returns the value of PH 10 to K:

K=PVAL (PH, 10).

An equivalent floating point example returns the value of PL5 to XK:
XK=FPVAL (PL, 5).

Errors in the argument lists of ASSIGN, LOGIC, FPVAL and PVAL cause branching to subroutine ARGERR, where statements indicating the problem nature are written. Upon return from ARGERR, the subroutine with the faulty argument list executes a no-op return to FORTRAN without interrupting the simulation. Three errors are recognized:

(1) An invalid parameter type referenced in calling ASSIGN, PVAL or FPVAL,

(2) An invalid switching operation specified in a call to LOGIC

(3) An attempt to assign a negative number to an integer parameter when calling ASSIGN.

RESTRICTIONS:

This subroutine branches to code internal to IBM GPSS-V in performing these functions. Use of any other system may produce unpredictable results from this subroutine.

PROGRAM LOGIC ASSIGN:

The program declares the aliases PVAL, FPVAL and LOGIC as entries at this subroutine. All registers except 13 are

saved and 12 is declared the base register for this subroutine. The save area address of the GPSS-V main program is obtained from the FORTRAN calling subprogram save area at the address contained in register 13 plus 4 bytes. Registers 2, 3, 10, and 11 of ASSIGN are loaded with the contents of the corresponding GPSS registers.

The constant stored at STPVAL is tested for zero to determine if ASSIGN has been called previously. A non-zero value causes branching to ASSIGNGO. For the zero value condition, the program obtains the addresses of the GPSS-V subroutines STPVAL and PRVAL. These addresses are stored at locations STPVAL and PRVAL respectively.

At ASSIGNGO, register 14 is loaded with contents of register 2 plus 4096 to fulfill a condition required for operation of STPVAL.

Register 10 is loaded with the address of GPSS-V control words from a 25 word table established by GPSS when the FORTRAN subprogram is called. The control word address will be used later to locate the number of the transaction currently being processed. Register 9 is loaded with the starting address of STPVAL.

Program location NEXTASGN is the beginning of a loop for processing the ASSIGN argument list. Register 1 initially contains the starting address of this list. Locations of the first three entries, which are parameter number, value and type, respectively, are loaded into registers 6 through 8. The address of the third entry is retained in register 0. Contents stored at the addresses contained in registers 6, 7, and 8 are loaded into these three respective registers.

A test for a floating point parameter is performed by loading the character stored at PL into register 4 and comparing this with the parameter type contained in register 8. The program branches to ASGNFLOT if a floating point parameter is present.

Before testing integer parameters for type, a test of the value to be assigned is required, because fixed point constants in GPSS block statements must not be negative. A test is performed on this value, which is contained in register 7. If a negative quantity is found, the program branches to NEGASSGN.

When the value is zero or positive, as normally expected, tests for halfword, fullword or byte parameters are performed by loading the characters stored at PH, PF or PB into register 4 and comparing these with the contents of register 8. The program branches to ASGNHALF, ASGNFULL or ASGNBYTE for each respective character type.

If none of the above three parameters are present in register 8, an error condition is recognized. The program places a value 1 in register 8 and continues to ASGERRET to begin an error indication procedure and subsequent return to the FORTRAN calling program.

This procedure requires branching to the subprogram ARGERR. At ASGERRET, the address of the FORTRAN calling program save area is loaded into register 10 from register 13. An ASSIGN save area of 18 fullwords starting at location SAVEAREA is defined. Register 13 is used as a linkage register and is loaded with the address of SAVEAREA. This address is also stored in the third word of the FORTRAN calling program and the address of the FORTRAN calling program is stored in the second word of SAVEAREA.

The error code value 1, contained in register 8, is stored at ERRCODE for use in the argument list when ARGERR is called. The address of the argument list ARGLIST, is loaded into register 1, the argument list linkage register, and the program branches to ARGERR. Upon return to ASSIGN, the address of the FORTRAN save area at SAVEAREA + 4 is loaded into register 13. Contents of registers 2 through 12 are restored to values contained when the FORTRAN subprogram called ASSIGN. The program branches back to the calling locations in the FORTRAN subprogram at the address contained in register 14.

For those parameter values previously tested and found to be negative, the program branched to NEGASSGN. AT this location an error code value of 8 is loaded into register 8. The program then branches back to ASGERRET to begin the error return procedure.

At ASGNHALF, ASGNFULL and ASGNFLOT, register 4 is loaded with the respective hexadecimal constants, 10000000, 0C000000 and 04000000, and a branch to MASKOP is executed. At ASGNBYTE, register 4 is loaded with the hexadecimal constant 08000000. The program continues to MASKOP.

The STPVAL entry conditions for register 6 are fulfilled by the OR statement at MASKOP. The transaction number is placed in register 8 and the program branches to STPVAL. Upon returning to ASSIGN, the program tests for the last argument list entry by examining the address stored in register 0 for a negative sign bit. If the end of the argument list is present, the program branches to RETASSGN.

The program continues processing the argument list by adding 12 to the contents in register 1 and branching back to NEXTASGN.

At RETASSGN the subprogram executes a normal return to the FORTRAN calling subprogram by executing a RETURN macro.

PROGRAM LOGIC:
PVAL and **FPVAL**

This section of the subroutine contains two entry points, PVAL and FPVAL. The FPVAL entry is located at the conclusion of PVAL. FPVAL establishes base registers, stores

the value 4096 at the storage location FLAG, then branches back to the location FORTSAVE in PVAL to begin processing the floating point parameter.

The PVAL section establishes register 12 as the base register. Zero is stored in the fullword location FLAG. At the instruction FORTSAVE, the program locates the FORTRAN save area, then loads registers 2, 3, 10 and 11 with the corresponding GPSS register contents, to prepare for the operation of GPSS subroutines STPVAL and PRVAL. The initial call to PVAL obtains the addresses of these two subroutines and stores them at locations STPVAL and PRVAL respectively. Subsequent calls test for a non-zero value at STPVAL and branch to PVALUEGO on this condition.

At PVALUEGO, register 14 is loaded with contents of register 2 plus 4096 to satisfy a GPSS condition for entry to STPVAL and PRVAL. The address of GPSS control words is loaded into register 10 for later use in determining the active transaction number. The addresses of the parameter number and type are loaded into registers 5 and 6 from the argument list address in register. The address of PRVAL is loaded into register 9.

The parameter number and type are loaded into registers 6 and 8 respectively from their storage locations. Register 8 contents are tested with the same characters as those in ASSIGN to determine parameter type. Branching to PHALF, PFULL, PFLOAT and PBYTE is executed for halfword, fullword, floating point and byte parameter types respectively. If none of these types are found, the program continues into an error return area. The error code is given a value 2 and the program executes instructions identical to those in the ASSIGN error return procedure.

At PHALF, PFULL and PFLOAT locations hexidecimal constants are loaded into register 4, then the program branches to MASKX. At PBYTE the program also loads a hexidecimal constant into register 5 and continues to MASKX. An OR instructions at MASKX places hexidecimal constants in bits 1-7 of register 6 for branching to subroutine PRVAL. The currently active transaction number is loaded into register 7.

The program branches to the PRVAL start location contained in register 9. Upon return, the value of FLAG is tested for a zero. If FLAG is non-zero, indicating a floating point parameter, the program branches to FLOATPT. For integer parameters, the program loads the parameter value returned from FPVAL in register 6 into result register 0, then branches to RETPVAL to initiate a procedure for returning to FORTRAN.

At FLOATPT, register zero is declared as a floating point register by an SDR instruction. The parameter value is first stored at VALUE, then loaded into result register 0. The program continues to RETPVAL.

At RETPVAL all registers except 0 and 1 are restored. The hexidecimal FF flag value is stored at the fourth word of the FORTRAN save area to indicate a return condition. The last program instruction location executes branching to the FORTRAN calling program return location.

PROGRAM LOGIC:

LOGIC

The LOGIC section establishes register 12 as a base register, then obtains the address of the GPSS save area from the FORTRAN save area. Contents of registers 3,10 and 11 from the GPSS save area are loaded into the respective LOGIC program registers. Register 10 contents, plus a displacement of 24 bytes, provide the address of GPSS control words which are subsequently loaded into register 10. A displacement of 1040 bytes beyond the control word address provides the starting address of the logic switches and this is placed in register 9.

The loop for performing logic switch setting and resetting begins at NXTLOGIC. At this location, addresses of the first two words of the argument list are loaded into registers 6 and 7 respectively. The address in register 7 is saved at LOGRFPTX. A logic set or reset halfword indicator and the logic switch number are loaded into registers 6 and 7 respectively, from the addresses contained in those two registers. The logic switch number is also placed in register 4. Register 7 is shifted left by 2 bits and register 4 by 1 bit. The addition of these in register 7 provides a multiplication by 6, the basic storage byte allocation for logic switches. This sum is also placed in register 4. Register 6 is examined to determine if a switch set or reset is to be implemented and branches to SET or RESET if the respective characters, LS or LR, are present. If the argument list is erroneous and contains neither character, the program assigns a value of 3 to the error code and implements procedures identical to the error routine coding in ASSIGN.

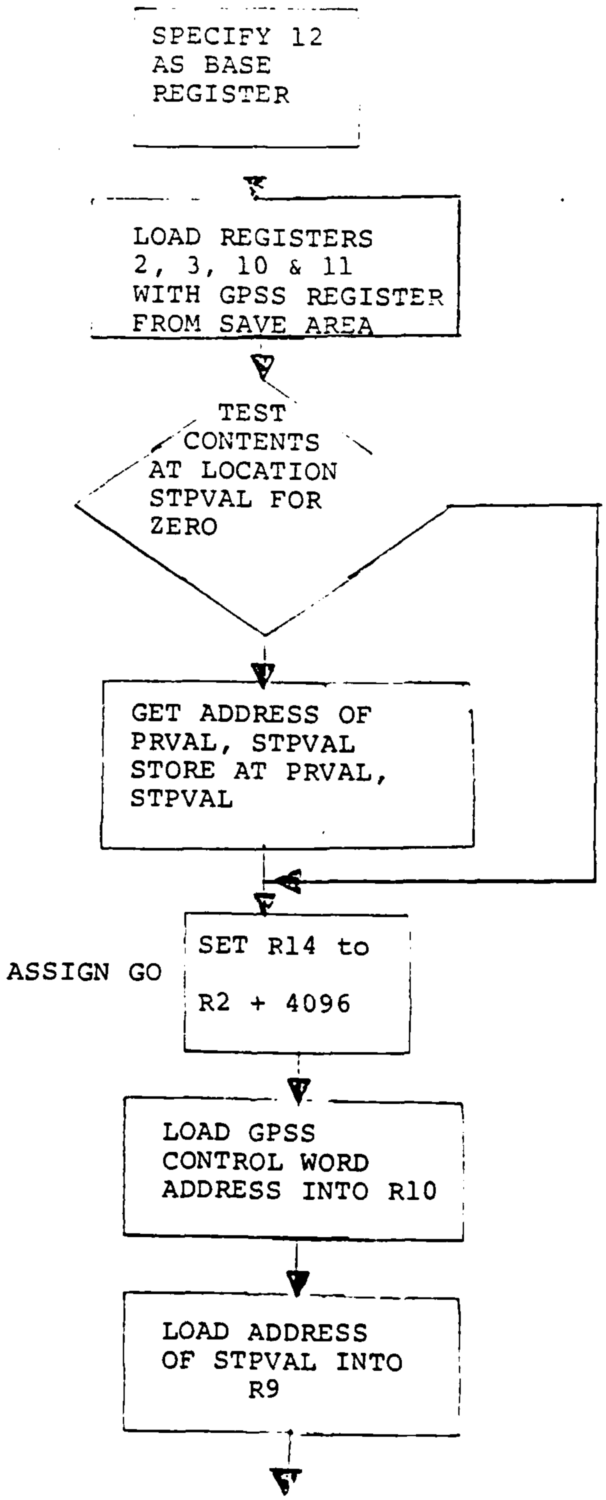
At RESET, register 0 is zeroed. A value of 4 is added to the quantity in register 4 (6 * switch number) and the program branches to SETRESET. A branch to SET loads the hexidecimal quantity 0014 into register 0, register 4 is incremented by 2 and the program continues to SETRESET.

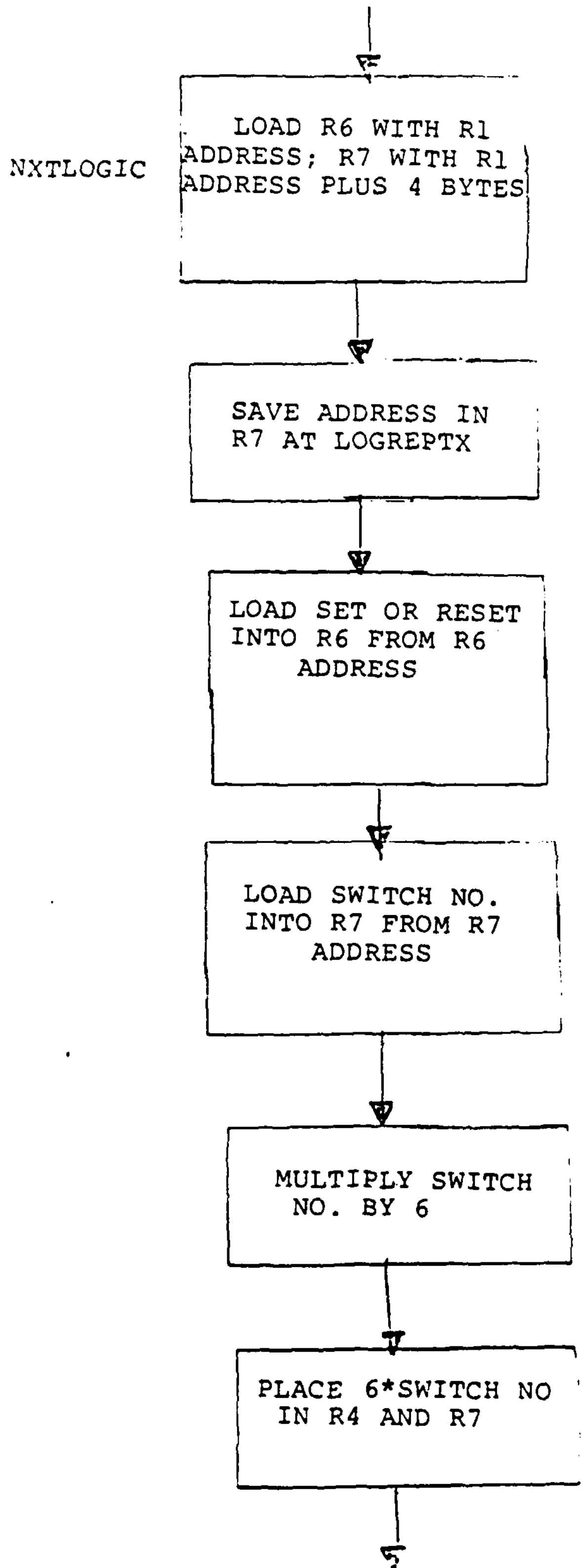
The indicator for a reset or set condition is contained in register 0. This halfword is stored in the first two bytes of the logic switch storage location by the instruction at SETRESET. The program branches to the GPSS chain maintenance area at 1688 bytes beyond the GPSS base address contained in

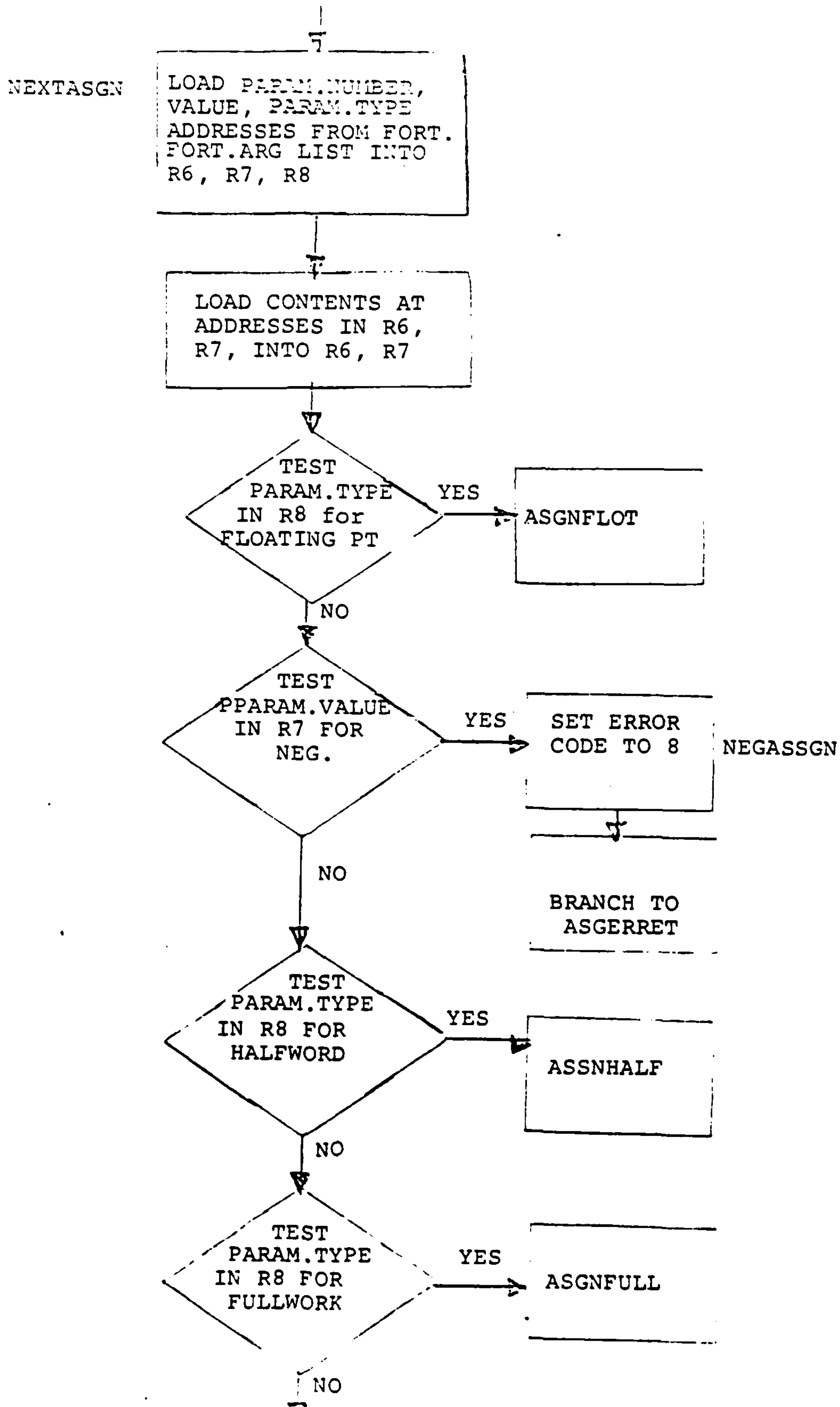
register 11. For a reset condition, the contents of bytes 5 and 6 are loaded into register 7. Register 8 contains the storage address of the chain holding transactions waiting for a reset condition. This address is stored at bytes 5 and 6. When a logic set is implemented, contents of bytes 3 and 4 are loaded into register 7. The storage location of the chain holding transactions waiting for a set condition is stored in bytes 3 and 4 from register 8.

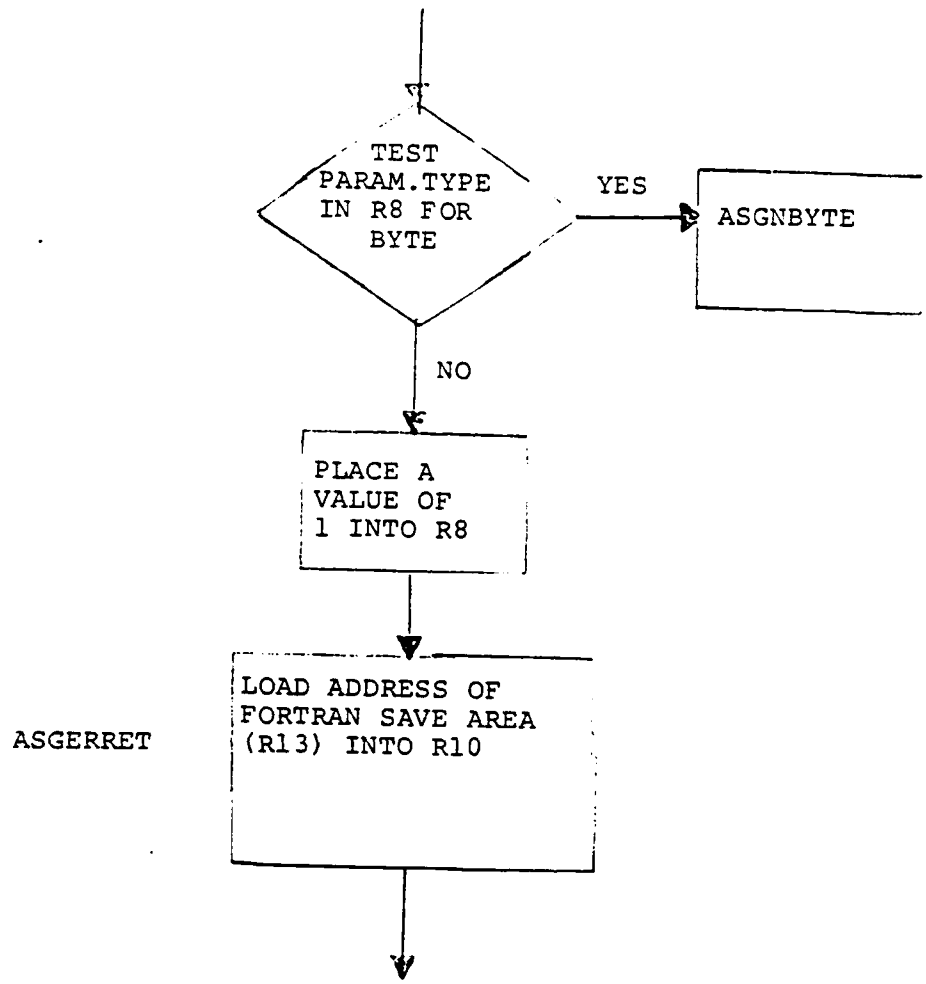
The argument list address stored at LOGREPTX is tested for a negative sign bit. When this occurs, the list is ended and the program branches to LOGICRET. The program continues by adding 8 to the contents of register 1 and branching back to NXTLOGIC. At LOGICRET the program executes the RETURN macro to return to the FORTRAN calling subprogram.

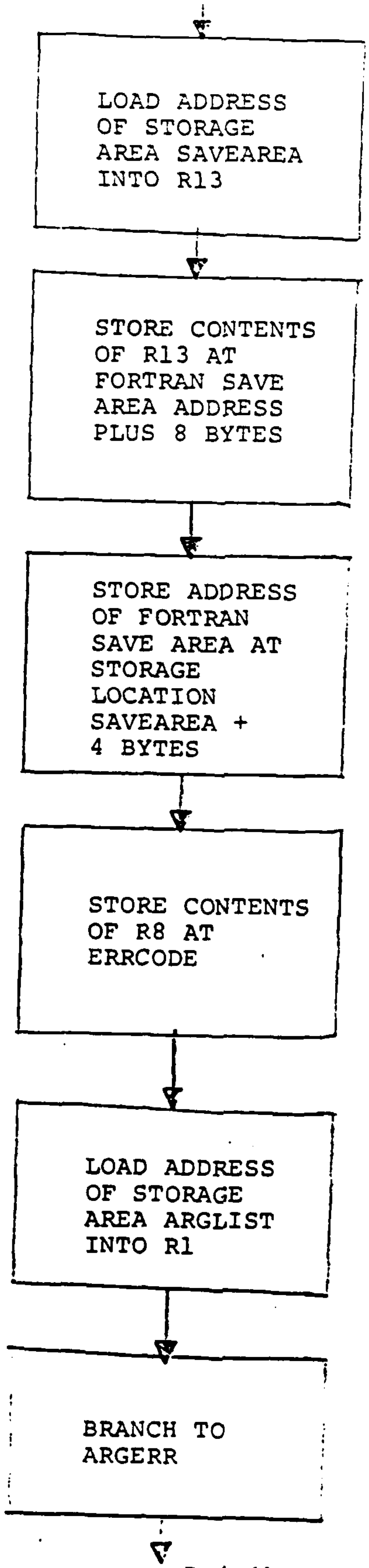
ASSIGN/LOGIC/PVAL/FPVAL

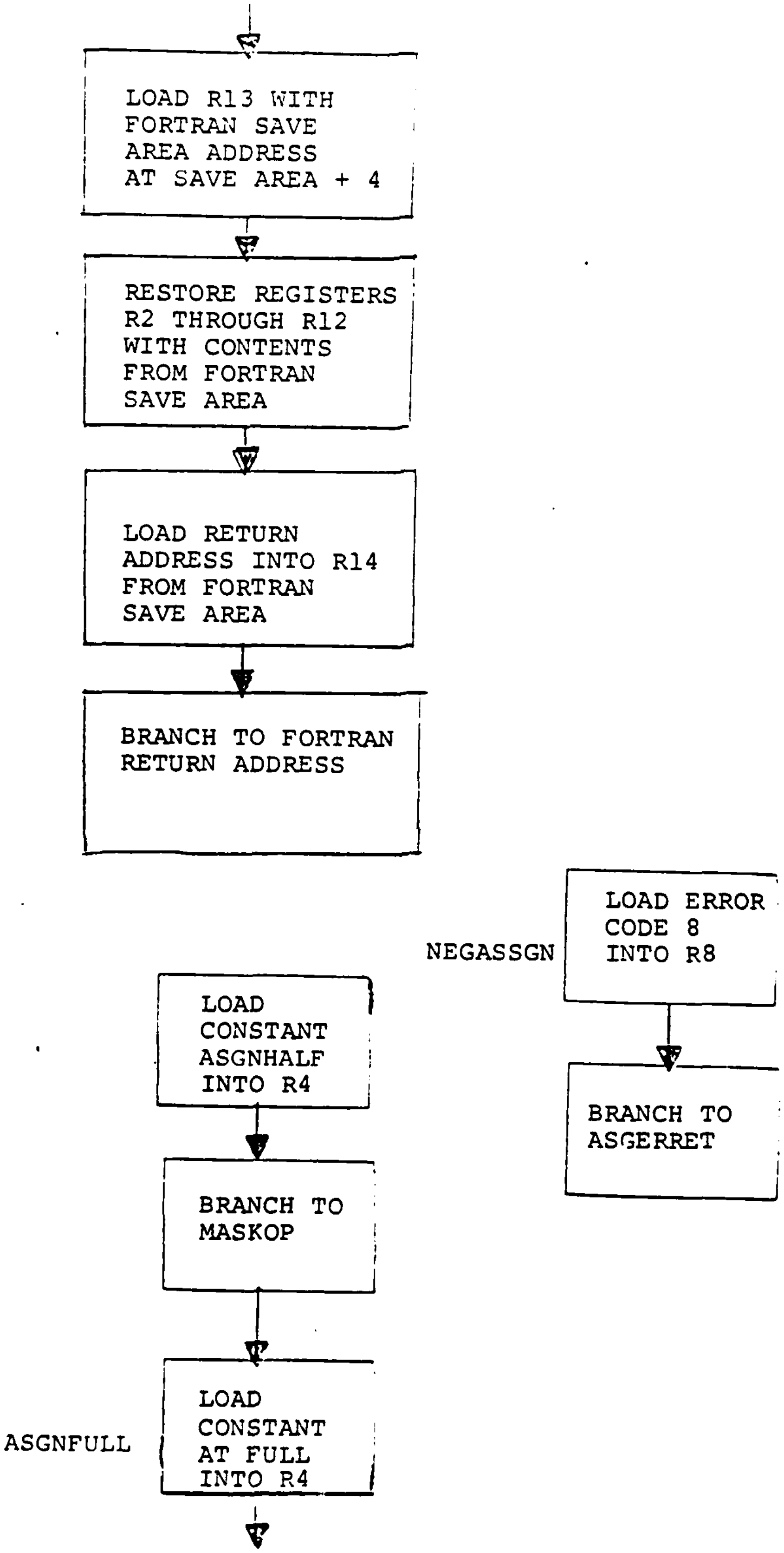


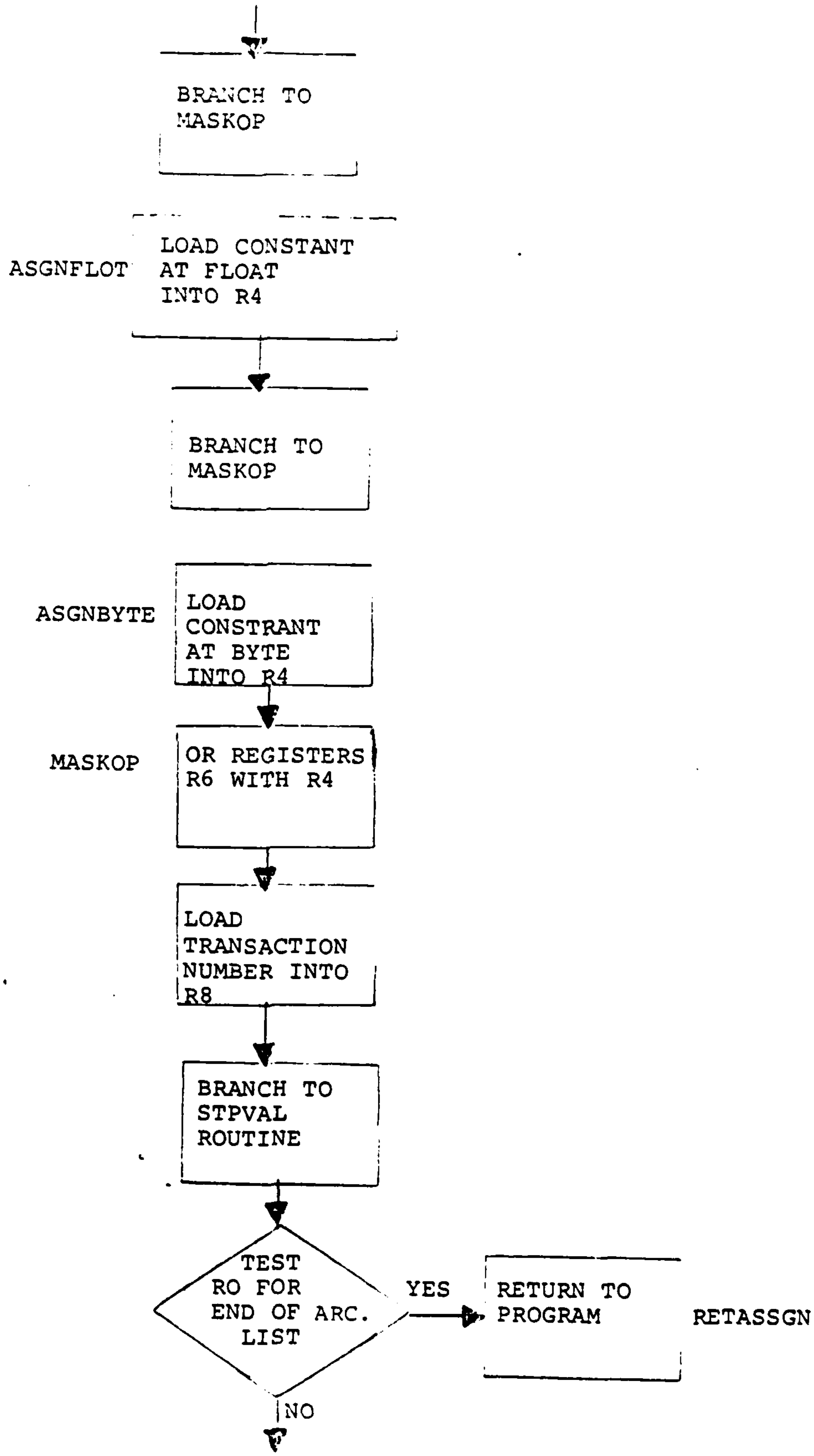

NXTLOGIC
LOAD R6 WITH R1 ADDRESS; R7 WITH R1 ADDRESS PLUS 4 BYTES
SAVE ADDRESS IN R7 AT LOGREPTX
LOAD SET OR RESET INTO R6 FROM R6 ADDRESS
LOAD SWITCH NO. INTO R7 FROM R7 ADDRESS
MULTIPLY SWITCH NO. BY 6
PLACE 6*SWITCH NO IN R4 AND R7

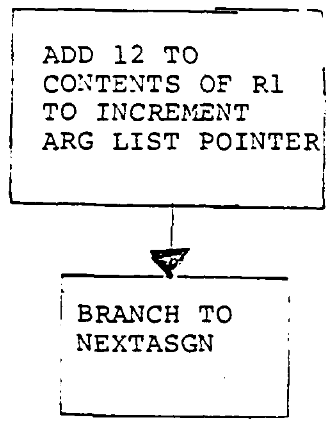

NEXTASGN
LOAD PARAM.NUMBER, VALUE, PARAM.TYPE ADDRESSES FROM FORT. FORT.ARG LIST INTO R6, R7, R8
LOAD CONTENTS AT ADDRESSES IN R6, R7, INTO R6, R7
TEST PARAM.TYPE IN R8 for FLOATING PT
YES
ASGNFLOT
NO
TEST PPARAM.VALUE IN R7 FOR NEG.
YES
SET ERROR CODE TO 8
NEGASSGN
BRANCH TO ASGERRET
NO
TEST PARAM.TYPE IN R8 FOR HALFWORD
YES
ASSNHALF
NO
TEST PARAM.TYPE IN R8 FOR FULLWORK
YES
ASGNFULL
NO

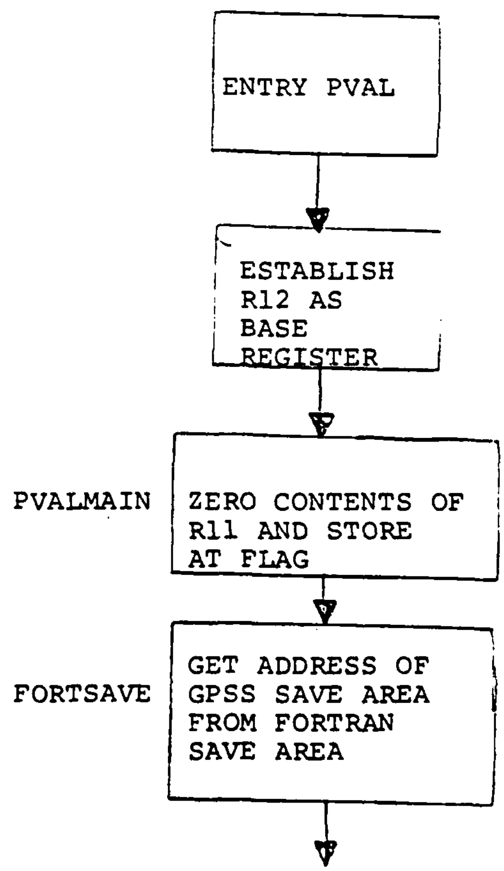

TEST PARAM.TYPE IN R8 FOR BYTE
YES
ASGNBYTE
NO
PLACE A VALUE OF 1 INTO R8
ASGERRET
LOAD ADDRESS OF FORTRAN SAVE AREA (R13) INTO R10

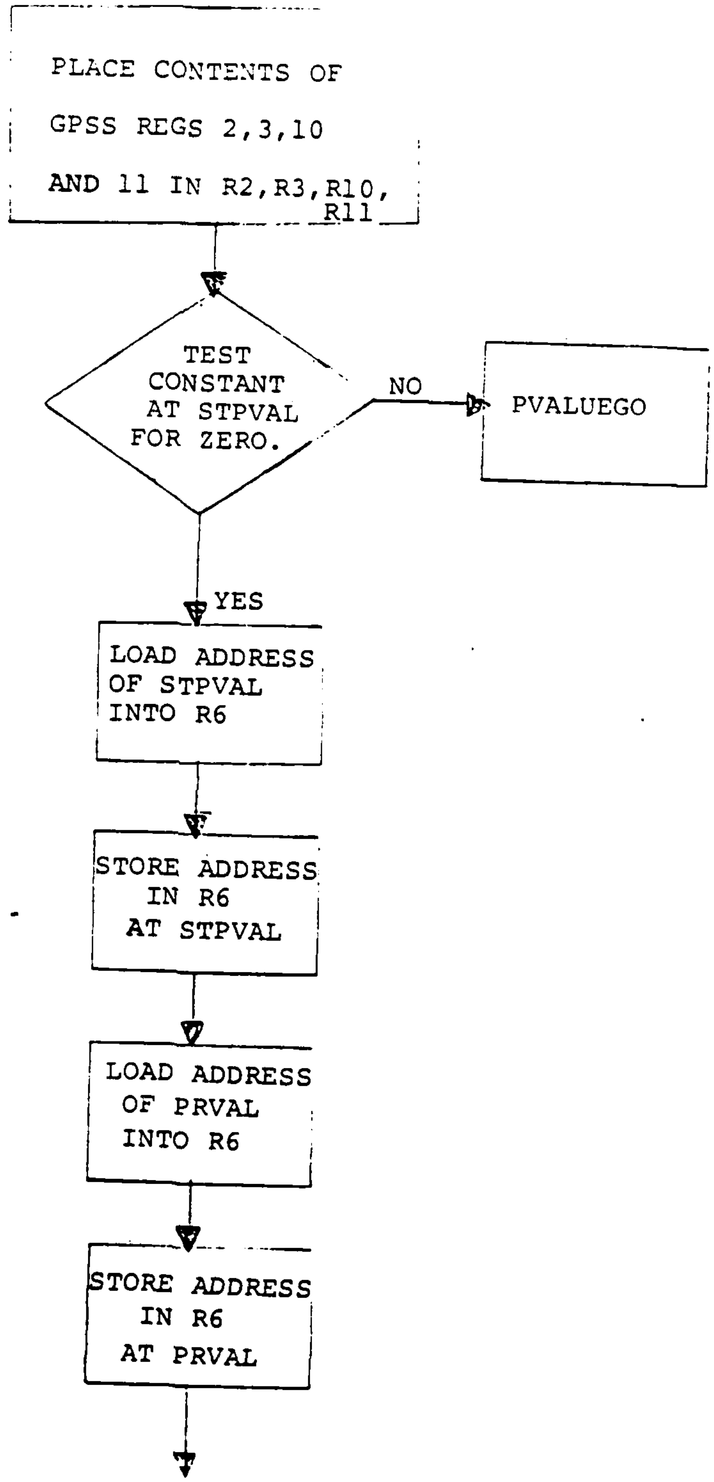

LOAD ADDRESS OF STORAGE AREA SAVEAREA INTO R13
STORE CONTENTS OF R13 AT FORTRAN SAVE AREA ADDRESS PLUS 8 BYTES
STORE ADDRESS OF FORTRAN SAVE AREA AT STORAGE LOCATION SAVEAREA + 4 BYTES
STORE CONTENTS OF R8 AT ERRCODE
LOAD ADDRESS OF STORAGE AREA ARGLIST INTO R1
BRANCH TO ARGERR

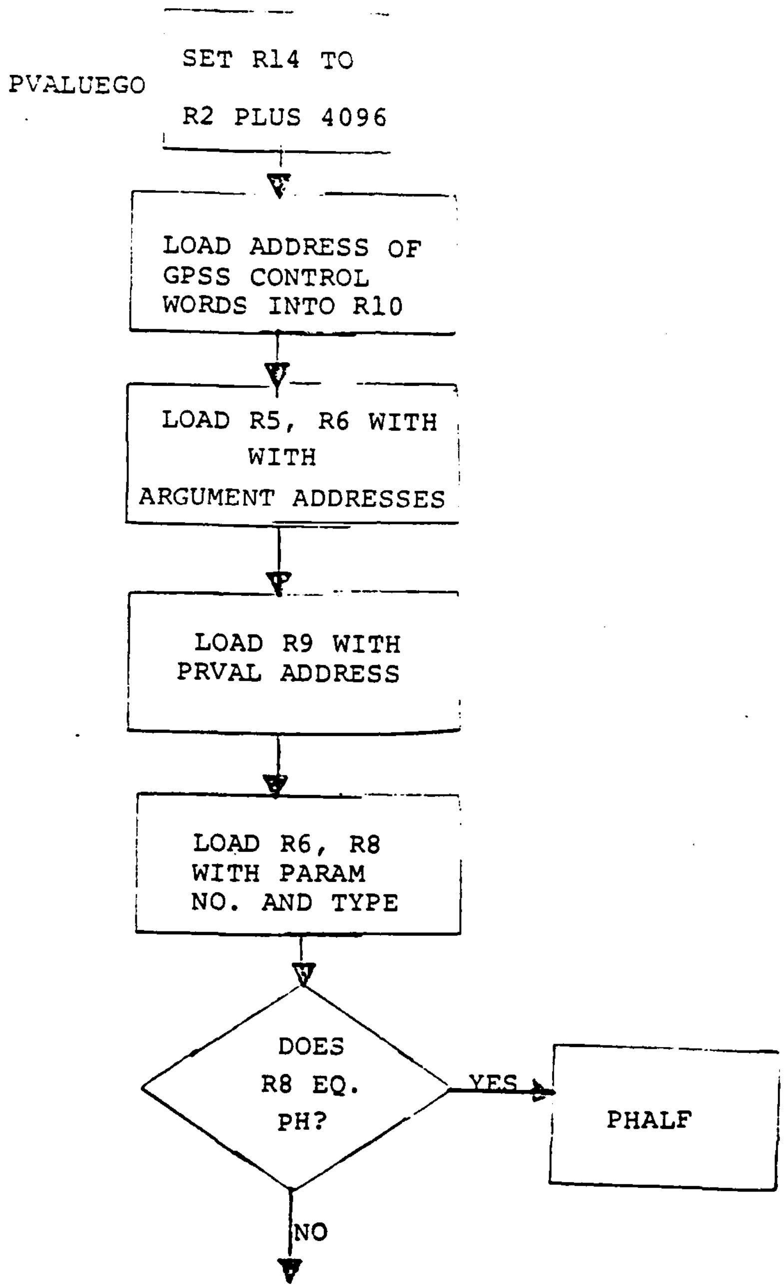

LOAD R13 WITH FORTRAN SAVE AREA ADDRESS AT SAVE AREA + 4
RESTORE REGISTERS R2 THROUGH R12 WITH CONTENTS FROM FORTRAN SAVE AREA
LOAD RETURN ADDRESS INTO R14 FROM FORTRAN SAVE AREA
BRANCH TO FORTRAN RETURN ADDRESS
NEGASSGN
LOAD ERROR CODE 8 INTO R8
BRANCH TO ASGERRET
LOAD CONSTANT ASGNHALF INTO R4
BRANCH TO MASKOP
ASGNFULL
LOAD CONSTANT AT FULL INTO R4

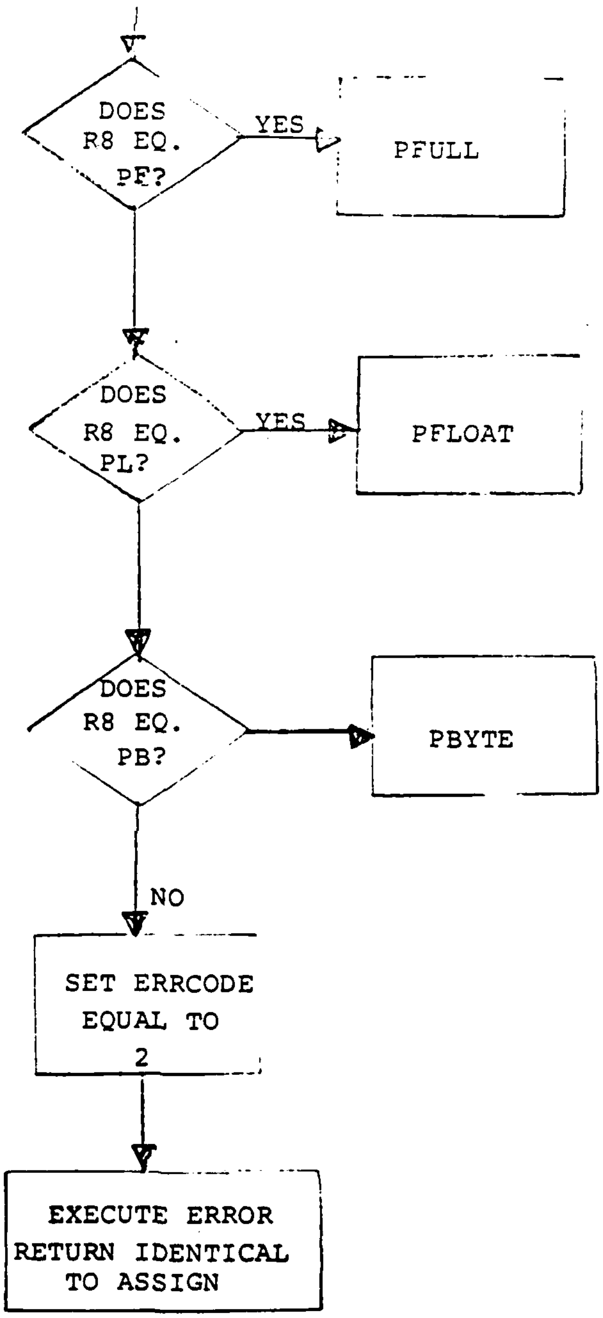

BRANCH TO MASKOP
ASGNFLOT
LOAD CONSTANT AT FLOAT INTO R4
BRANCH TO MASKOP
ASGNBYTE
LOAD CONSTRANT AT BYTE INTO R4
MASKOP
OR REGISTERS R6 WITH R4
LOAD TRANSACTION NUMBER INTO R8
BRANCH TO STPVAL ROUTINE
TEST RO FOR END OF ARC. LIST
YES
RETURN TO PROGRAM
RETASSGN
NO

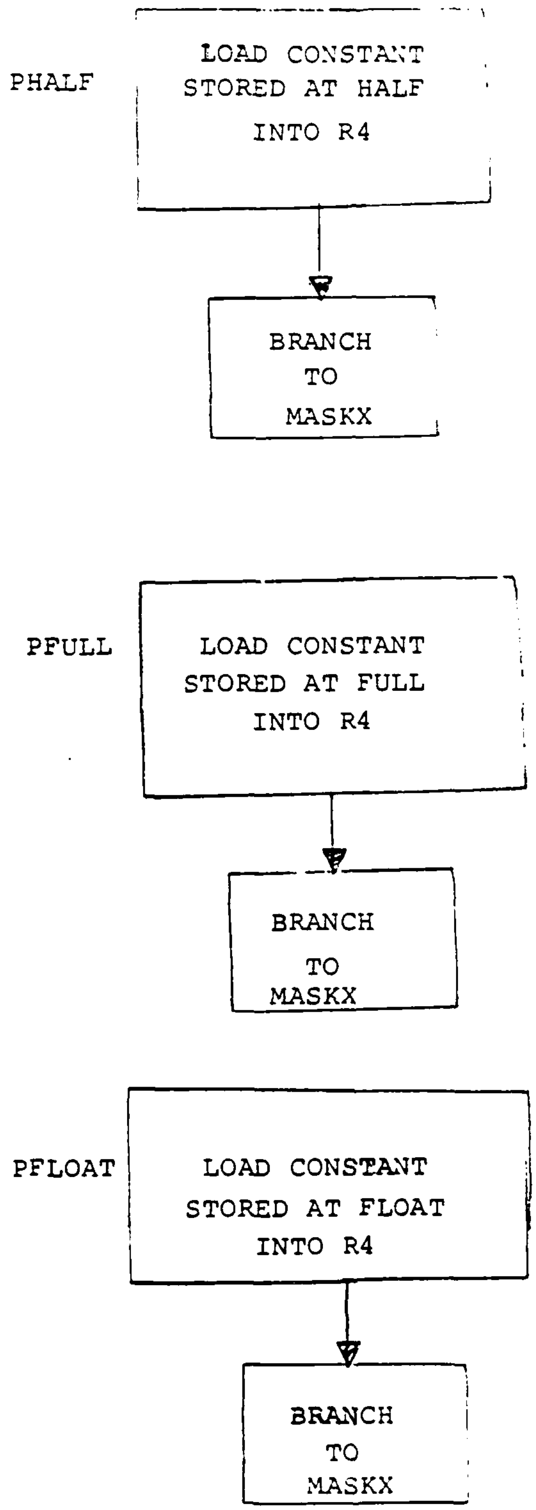

ADD 12 TO CONTENTS OF R1 TO INCREMENT ARG LIST POINTER
BRANCH TO NEXTASGN

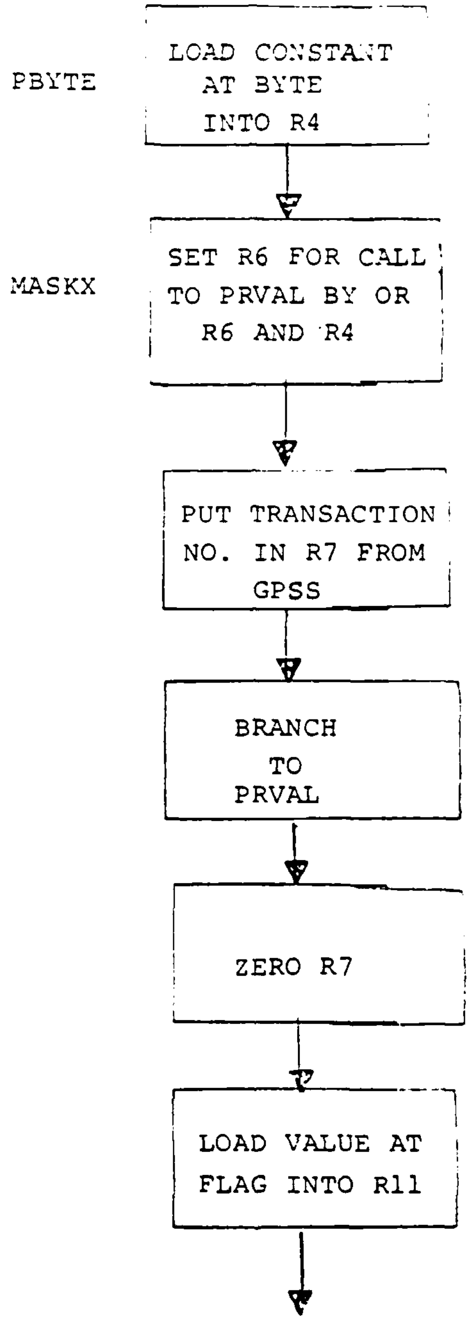

ENTRY PVAL
ESTABLISH R12 AS BASE REGISTER
PVALMAIN
ZERO CONTENTS OF R11 AND STORE AT FLAG
FORTSAVE
GET ADDRESS OF GPSS SAVE AREA FROM FORTRAN SAVE AREA


PLACE CONTENTS OF GPSS REGS 2,3,10 AND 11 IN R2,R3,R10, R11
TEST CONSTANT AT STPVAL FOR ZERO.
NO
PVALUEGO
YES
LOAD ADDRESS OF STPVAL INTO R6
STORE ADDRESS IN R6 AT STPVAL
LOAD ADDRESS OF PRVAL INTO R6
STORE ADDRESS IN R6 AT PRVAL


PVALUEGO
SET R14 TO
R2 PLUS 4096
LOAD ADDRESS OF
GPSS CONTROL
WORDS INTO R10
LOAD R5, R6 WITH
WITH
ARGUMENT ADDRESSES
LOAD R9 WITH
PRVAL ADDRESS
LOAD R6, R8
WITH PARAM
NO. AND TYPE
DOES
R8 EQ.
PH?
YES
PHALF
NO


DOES
R8 EQ.
PF?
YES
PFULL
DOES
R8 EQ.
PL?
YES
PFLOAT
DOES
R8 EQ.
PB?
PBYTE
NO
SET ERRCODE
EQUAL TO
2
EXECUTE ERROR
RETURN IDENTICAL
TO ASSIGN


PHALF
LOAD CONSTANT STORED AT HALF INTO R4
BRANCH TO MASKX
PFULL
LOAD CONSTANT STORED AT FULL INTO R4
BRANCH TO MASKX
PFLOAT
LOAD CONSTANT STORED AT FLOAT INTO R4
BRANCH TO MASKX

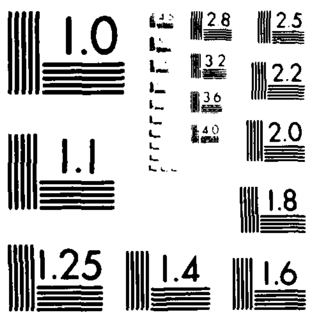

PBYTE
LOAD CONSTANT AT BYTE INTO R4
MASKX
SET R6 FOR CALL TO PRVAL BY OR R6 AND R4
PUT TRANSACTION NO. IN R7 FROM GPSS
BRANCH TO PRVAL
ZERO R7
LOAD VALUE AT FLAG INTO R11

AD-A117 603 TRANSPORTATION SYSTEMS CENTER CAMBRIDGE MA F/G 1/5
AIRPORT LANDSIDE. VOLUME V. APPENDIX B. ALSIM SUBROUTINES.(U)
JUN 82 L MCCABE, M GORSTEIN
UNCLASSIFIED DOT-TSC-FAA-82-4-5 FAA-EM-80-8-5 NL

4 OF 4

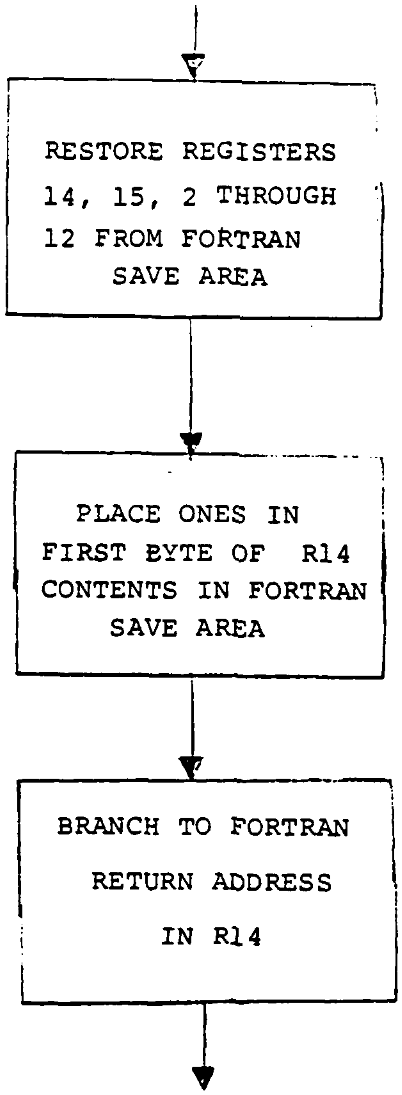
AD A117603

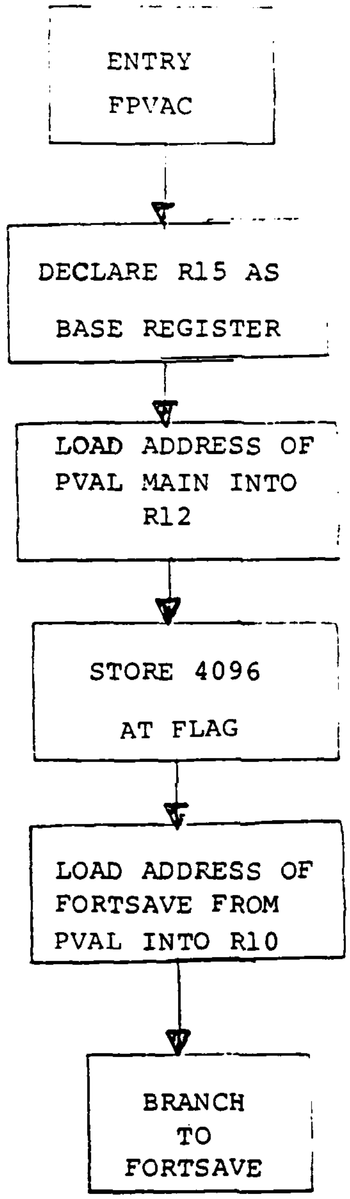
END
DATE FILMED
9 82
DTIC

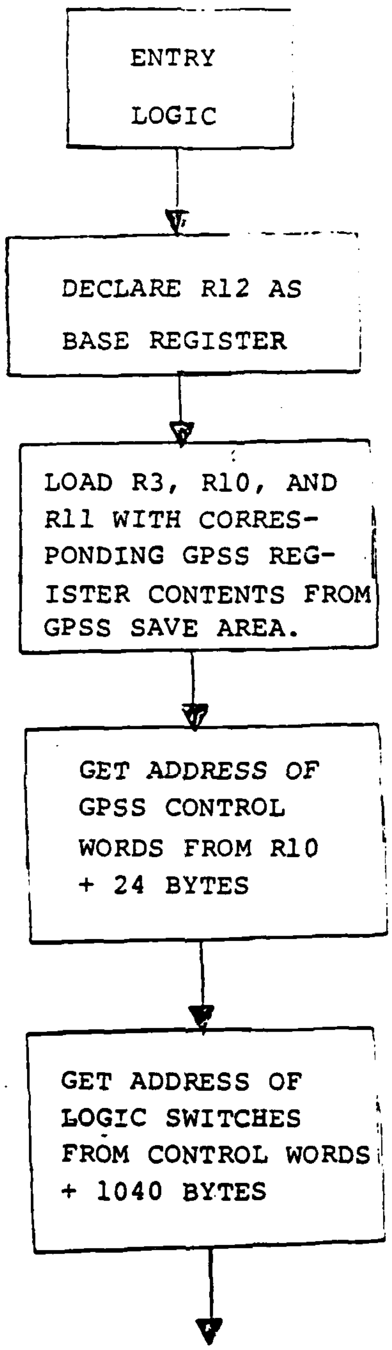


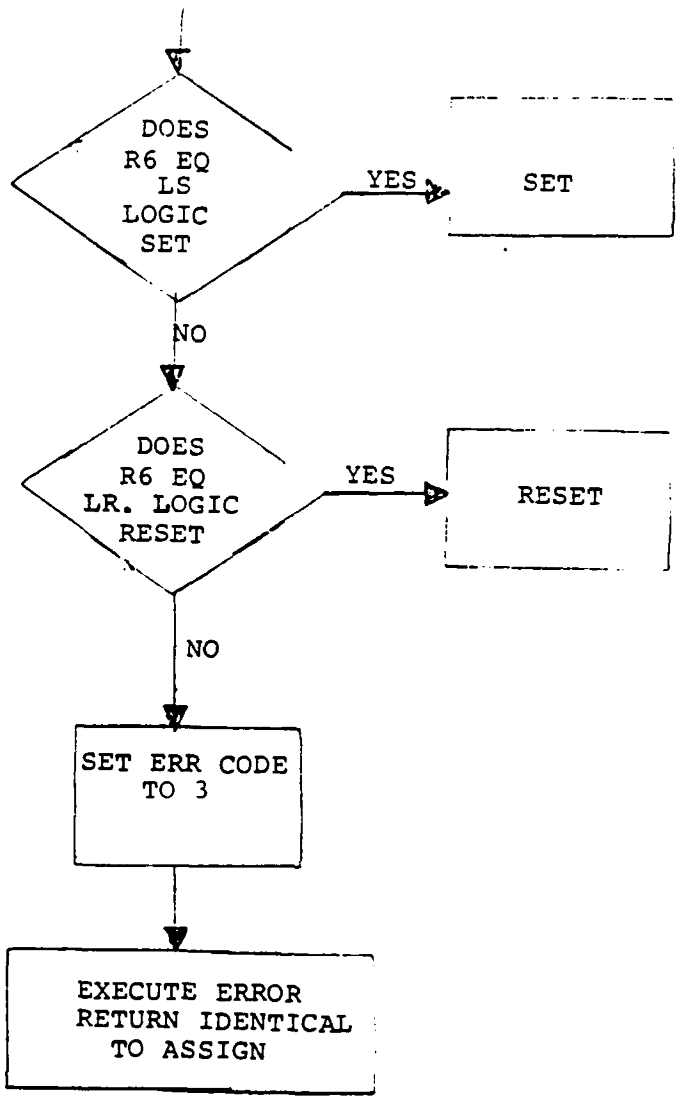

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

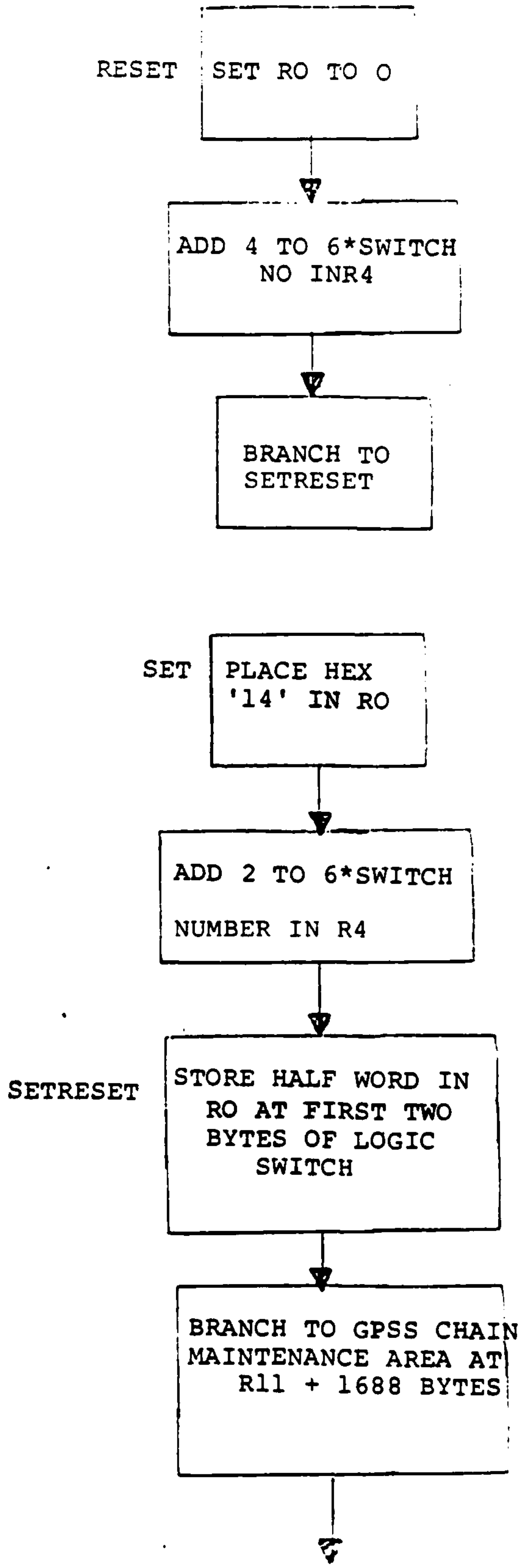
RETPVAL

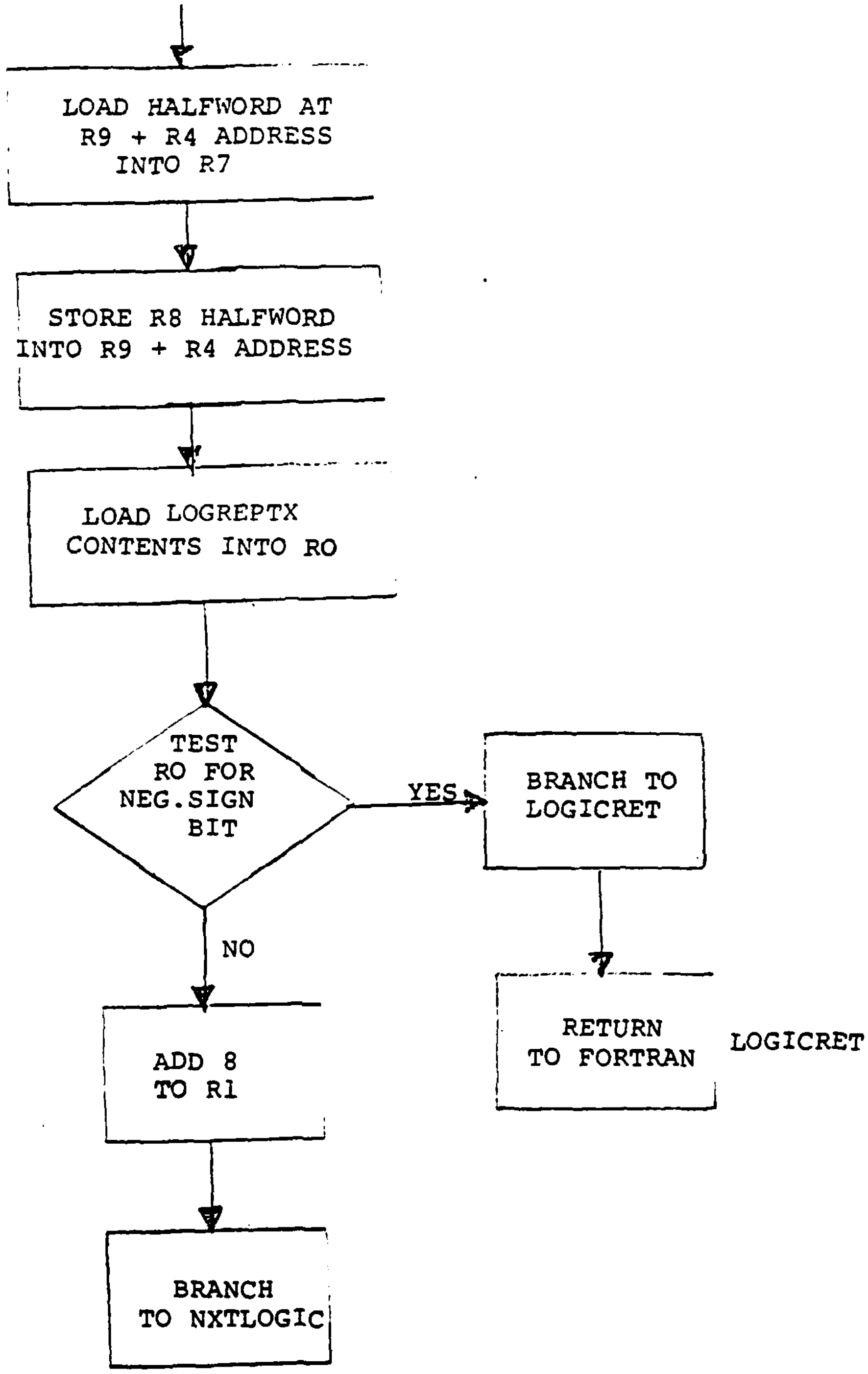

RESTORE REGISTERS
14, 15, 2 THROUGH
12 FROM FORTRAN
SAVE AREA
PLACE ONES IN
FIRST BYTE OF R14
CONTENTS IN FORTRAN
SAVE AREA
BRANCH TO FORTRAN
RETURN ADDRESS
IN R14


ENTRY
FPVAC
DECLARE R15 AS
BASE REGISTER
LOAD ADDRESS OF
PVAL MAIN INTO
R12
STORE 4096
AT FLAG
LOAD ADDRESS OF
FORTSAVE FROM
PVAL INTO R10
BRANCH
TO
FORTSAVE


ENTRY
LOGIC
DECLARE R12 AS
BASE REGISTER
LOAD R3, R10, AND
R11 WITH CORRES-
PONDING GPSS REG-
ISTER CONTENTS FROM
GPSS SAVE AREA.
GET ADDRESS OF
GPSS CONTROL
WORDS FROM R10
+ 24 BYTES
GET ADDRESS OF
LOGIC SWITCHES
FROM CONTROL WORDS
+ 1040 BYTES


DOES
R6 EQ
LS
LOGIC
SET
YES
SET
NO
DOES
R6 EQ
LR. LOGIC
RESET
YES
RESET
NO
SET ERR CODE
TO 3
EXECUTE ERROR
RETURN IDENTICAL
TO ASSIGN


RESET
SET RO TO O
ADD 4 TO 6*SWITCH NO INR4
BRANCH TO SETRESET
SET
PLACE HEX '14' IN RO
ADD 2 TO 6*SWITCH NUMBER IN R4
SETRESET
STORE HALF WORD IN RO AT FIRST TWO BYTES OF LOGIC SWITCH
BRANCH TO GPSS CHAIN MAINTENANCE AREA AT R11 + 1688 BYTES


LOAD HALFWORD AT R9 + R4 ADDRESS INTO R7
STORE R8 HALFWORD INTO R9 + R4 ADDRESS
LOAD LOGREPTX CONTENTS INTO RO
TEST RO FOR NEG.SIGN BIT
YES
BRANCH TO LOGICRET
RETURN TO FORTRAN
LOGICRET
NO
ADD 8 TO R1
BRANCH TO NXTLOGIC

```
MEMBER NAME  ASSIGN
*   A S S I G N
*
M   THIS IS A PACKAGE OF FORTRAN CALLABLE SUBROUTINES AND FUNCTIONS
*   TO PERMIT GPSS-FORTRAN COMMUNICATION NOT SUPPORTED IN THE
*   IBM GPSS-V PACKAGE.   THEY ARE:
*        OBTAIN A VALUE FROM A PARAMETER OF THE ACTIVE TRANSACTION
*        ASSIGN A VALUE TO A PARAMETER OF THE ACTIVE TRANSACTION
*        SET OR RESET A LOGIC SWITCH
*
*
*
*   TO USE ASSIGN FEATURE:
*                         CALL ASSIGN(PARAMETER,VALUE,TYPE)
*                         WHERE TYPE IS PF,PH,PL OR PB.
*                         ANY OTHER TYPE CODE RESULTS IN A CALL TO
*                         ARGERR AND A NO-OP RETURN TO CALLING HELP
*                         BLOCK.
*
*   GPSS - ASSIGN   1,10,PH
*   ---> - CALL ASSIGN(1,10,PH)
*
*                         NOTE:   MULTIPLE ASSIGNMENTS VALID.
*                                 CALL ASSIGN(1,10,PH,1,100,PF)
*                                 RELPACES 2 ASSIGN BLOCKS.
*
*
*   TO USE LOGIC FEATURE:
*                         CALL LOGIC(LS/LR,SWITCH NUMBER)
*                         ANY CODE OTHER THAN LR OR LS RESULTS IN NO-OP
*
*   GPSS - LOGIC S   5
*   ---> - CALL LOGIC(LS,5)
*
```

```
*                        NOTE:   MULTIPLE SET/RESETS VALID.
*                                CALL LOGIC(LS,1,LS,3,LR,4)
*                                REPLACES 3 LOGIC BLOCKS.
*
*
*   TO REFERENCE INTEGER PARAMETER VALUE:
*                        PVAL(TYPE,PARAMETER)
*                        WHERE TYPE IS PF,PH OR PB.
*                        ANY OTHER TYPE CODE RESULTS IN A NO-OP.
*
*   EXAMPLE:  K=PVAL(PH,10) RETURNS PH10 TO K.
*
*
*     NOTE!   MUST BE ASSIGNED ALIASES OF LOGIC, PVAL, AND FPVAL.
*
*
ASSIGN   START 0
         ENTRY PVAL,FPVAL,LOGIC
         B     12(15)
         DC    X'7'
         DC    CL7'ASSIGN '
         SAVE  (14,12)
         BALR  12,0
         USING *,12
         L     5,4(13)          PT TO FORT SAVE
         LM    2,3,28(5)        GPSSREG 2-3
         LM    10,11,60(5)      GPSSREG 10-11
```

```
         SR    5,5
         L     6,STPVAL
         CR    5,6
         BNE   ASSIGNGO
         L     6,52(10)         ADDR OF STPVAL
         ST    6,STPVAL
         L     6,60(10)         ADDR OF PRVAL
         ST    6,PRVAL
ASSIGNGO LR    14,2
         A     14,=F'4096'
         L     10,24(10)        GPSS CONTROL WORDS
         L     9,STPVAL         STPVAL ADDR
NEXTASGN LM    6,8,0(1)         FORT ARG LIST
         LR    0,8              SAVE LAST ADDR IN ARG LIST
         L     6,0(6)           PARM NUMBER
         L     7,0(7)           VALUE
         LH    8,0(8)           PARM TYPE
         LH    4,PL             FLOATING TEST
         CR    4,8
         BE    ASGNFLOT
         LTR   7,7              TEST FOR NEG VALUE
         BC    4,NEGASSGN
         LH    4,PH             HALFWORD TEST
         CR    4,8
         BE    ASGNHALF
         LH    4,PF             FULLWORD TEST
         CR    4,8
         BE    ASGNFULL
         LH    4,PB             BYTE TEST
         CR    4,8
         BE    ASGNBYTE
         LA    8,1              ERROR CODE
ASGERRET LR    10,13            FORT SAVE AREA
         LA    13,SAVEAREA
         ST    13,8(0,10)       BACKWARD SAVE CHAIN
         ST    10,4(0,13)       FORWARD SAVE CHAIN
         ST    8,ERRCODE
         LA    1,ARGLIST
         L     15,=V(ARGERR)
         BALR  14,15
         L     13,SAVEAREA+4
         LM    2,12,28(13)      ERROR - RETURN
```

```
         L     14,12(13)        RETURN ADDR
         BR    14
NEGASSGN LA    8,8              ERROR CODE
         B     ASGERRET         BRANCH BACK TO ERROR COND RET
ASGNHALF L     4,HALF
         B     MASKOP
ASGNFULL L     4,FULL
         B     MASKOP
ASGNFLOT L     4,FLOAT
         B     MASKOP
ASGNBYTE L     4,BYTE
MASKOP   OR    6,4
         LH    6,738(10)        XAC NO
         BALR  5,9
         LTR   0,0
         BC    4,RETASSGN
         A     1,=F'12'
         B     NEXTASGN
RETASSGN RETURN (14,12)

         DS    0F
PVAL     B     10(15)
         DC    CL5'PVAL '
         DC    X'5'
         SAVE  (14,12)
         BALR  12,0
         USING *,12
PVALMAIN SR    11,11            SET POS BY FLOATING PT PARAMETER
         ST    11,FLAG
FORTSAVE L     5,4(13)          PT TO FORT SAVE
         LM    2,3,28(5)        GPSSREG 2-3
         LM    10,11,60(5)      GPSSREG 10-11
         SR    5,5
         L     6,STPVAL
```

```
         CR    5,6
         BNE   PVALUECO
         L     6,52(10)       ADDR OF STPVAL
         ST    6,STPVAL
         L     6,60(10)       ADDR OF PRVAL
         ST    6,PRVAL
PVALUEGO LR    14,2
         A     14,=F'4096'
         L     10,24(10)      GPSS CONTROL WORDS
         LM    5,6,0(1)       FORT ARG LIST
         L     9,PRVAL        PRVAL ADDR
         L     6,0(6)         PARM NUMBER
         LH    8,0(5)         PARM TYPE
         LH    4,PH           HALFWORD TEST
         CR    4,8
         BE    PHALF
         LH    4,PF           FULLWORD TEST
         CR    4,8
         BE    PFULL
         LH    4,PL           FLOATING TEST
         CR    4,8
         BE    PFLOAT
         LH    4,PB           BYTE TEST
         CR    4,8
         BE    PBYTE
         LR    10,13          FORT SAVE AREA
         LA    13,SAVEAREA
         ST    13,8(0,10)     BACKWARD SAVE CHAIN
         ST    10,4(0,13)     FORWARD SAVE CHAIN
         LA    8,2            ERROR CODE
         ST    8,ERRCODE
         LA    1,ARGLIST
         L     15,=V(ARGERR)
         BALR  14,15
         L     13,SAVEAREA+4
         LM    2,12,28(13)    ERROR - RETURN
```

```
         L     14,12(13)     RETURN ADDR                                  00173000
         BR    14                                                         00174000
PHALF    L     4,HALF                                                     00175000
         B     MASKX                                                      00176000
PFULL    L     4,FULL                                                     00177000
         B     MASKX                                                      00178000
PFLOAT   L     4,FLOAT                                                    00179000
         B     MASKX                                                      00180000
PBYTE    L     4,BYTE                                                     00181000
MASKX    OR    6,4                                                        00182000
         IC    7,738(10)     XAC NO                                       00183000
         BALR  5,9                                                        00184000
         SR    7,7                                                        00185000
         L     11,FLAG                                                    00186000
         CR    7,11                                                       00187000
         BNE   FLOATPT                                                    00188000
         LR    0,6                                                        00189000
         B     RETPVAL                                                    00190000
FLOATPT  SDR   0,0                                                        00191000
         ST    6,VALUE                                                    00192000
         LE    0,VALUE                                                    00193000
RETPVAL  LM    14,15,12(13)                                               00194000
         LM    2,12,28(13)                                                00195000
         MVI   12(13),X'FF'                                               00196000
         BR    14                                                         00197000
         DS    0F                                                         00198000
FPVAL    B     10(15)                                                     00199000
         DC    CL5'FPVAL'                                                 00200000
         SAVE  (14,12)                                                    00201000
         DROP  12                                                         00202000
         USING FPVAL,15                                                   00203000
         L     12,=A(PVALMAIN)                                            00204000
```

```
         DROP  15
         USING PVALMAIN,12
         L     11,=F'4096'
         ST    11,FLAG
         L     10,=A(FORTSAVE)
         BR    10
         DS    0F
LOGIC    B     10(15)
         DC    CL5'LOGIC'
         SAVE  (14,12)
         BALR  12,0
         USING *,12
         L     2,4(13)
         L     3,32(2)         GPSS R3
         LM    10,11,60(2)     GPSS R10,BASE
         L     10,24(10)       GPSS CONTROL
         L     9,1040(10)      GPSS ILOG
NXTLOGIC LM    6,7,0(1)
         ST    7,LOGREPTX      SAVE LAST ADDR OF ARG LIST
         LH    6,0(6)          LS OR LR
         L     7,0(7)          SWITCH NO
         LR    4,7
         SLA   7,2
         SLA   4,1
         AR    7,4             6 * SWITCH NO.
         LR    4,7
         LH    5,LS
         CR    5,6
         BE    SET
         LH    5,LR
         CR    5,6
         BE    RESET
         LR    10,13           FORT SAVE AREA
         LA    13,SAVEAREA
         ST    13,8(0,10)      BACKWARD SAVE CHAIN
         ST    10,4(0,13)      FORWARD SAVE CHAIN
         LA    8,3             ERROR CODES
         ST    8,ERRCODE
         LA    1,ARGLIST
         L     15,=V(ARGERR)
```

```
         BALR  14,15
         L     13,SAVEAREA+4
         LM    2,12,28(13)      ERROR - RETURN
         L     14,12(13)        RETURN ADDR
         BR    14
RESET    SR    0,0
         A     4,=F'4'
         B     SETRESET
SET      LH    0,=X'0014'
         A     4,=F'2'
SETRESET STH   0,0(7,9)
         BAL   5,1683(11)
         LH    7,0(9,4)
         STH   8,0(9,4)
         L     0,LOGREPTX
         LTR   0,0
         BC    4,LOGICRET
         A     1,=F'8'
         B     NXTLOGIC
LOGICRET RETURN (14,12)
SAVEAREA DS    18F
ERRCODE  DS    1F
ARGLIST  DC    X'60'
```

```
         DC    AL3(ERRCODE)
LOGREPTX DS    1F
PRVAL    DS    1F
STPVAL   DC    X'00000000'
FLAG     DS    1F
VALUE    DS    1F
MASK     DC    1X'00FFFFFF'
ADCON1   DC    A(FORTSAVE)
ADCON2   DC    A(PVALMAIN)
BYTE     DC    X'08000000'
HALF     DC    X'10000000'
FULL     DC    X'0C000000'
FLOAT    DC    X'04000000'
PB       DC    CL2'PB'
PH       DC    CL2'PH'
PF       DC    CL2'PF'
PL       DC    CL2'PL'
LS       DC    CL2'LS'
LR       DC    CL2'LR'
         DC    X'2'
          END
```

## FORTRAN SUBROUTINE ARGERR

PURPOSE:

This subroutine displays messages when errors in the argument lists of assembler subprograms ASSIGN, LOGIC, FPVAL and PVAL are detected. Recognized errors are:

1. An invalid parameter type referenced in calling ASSIGN, PVAL or FPVAL.

2. An invalid switching operation specified in a call to LOGIC.

3. An attempt to assign a negative number to an integer parameter when calling ASSIGN.

USAGE:

This subroutine is link edited with the name ARGERR. Subroutines ASSIGN, LOGIC, PVAL and FPVAL call this subroutine using the argument ERCODE if the above errors occur. When an invalid parameter type occurs in ASSIGN, a value of 1 is stored at ERRCODE, then ASSIGN branches to ARGERR with this argument value. Invalid parameter types in PVAL and FPVAL both produce an ERRCODE value of 2 before branching to ARGERR. Calls to LOGIC with invalid switching operations specified make ERRCODE equal to 3 and, lastly, attempts to assign negative numbers to integer parameters in ASSIGN are given an ERRCODE value of 8.

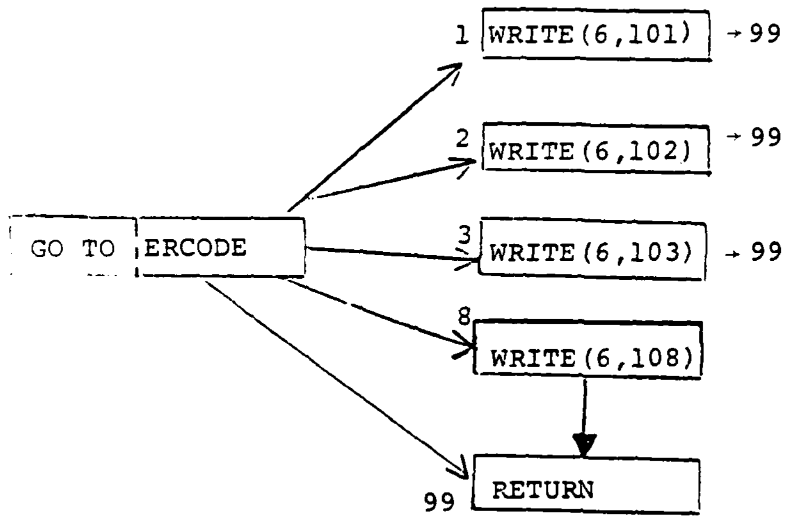
RESTRICTIONS:

None

PROGRAM LOGIC:

The program executes a computed GO TO statement and branches to the appropriate WRITE statement for the condition specified by the argument ERCODE. After execution of the WRITE statement, the program returns to ASSIGN, LOGIC, FPVAL and PVAL

SUBROUTINE ARGERR(ERCODE)



```
MEMBER NAME  ARGERR
      SUBROUTINE ARGERR(ERCODE)
C
C     SUBROUTINE RUN AS COMPANION TO ASSIGN SUBROUTINE.   PRINTS ERROR
C     MESSAGE WHEN INVALID PARAMETER TYPE SPECIFIED IN CALL TO ASSIGN,
C     PVAL, OR LOGIC.
C
      INTEGER*4 ERCODE
      GOTO(1,2,3,99,99,99,99,8),ERCODE
    1 WRITE(6,101)
  101 FORMAT(///,' *****  SPECIFIED INVALID PARAMETER TYPE IN ASSIGN CAL
     -L.   PROGRAM CONTINUES.'///)
      GOTO 99
    2 WRITE(6,102)
  102 FORMAT(///,' *****  SPECIFIED INVALID PARAMETER TYPE IN PVAL CALL.
     -   PROGRAM CONTINUES.',///)
      GOTO 99
    3 WRITE(6,103)
  103 FORMAT(///,' *****  SPECIFIED INVALID SWITCHING OPERATION IN LOGIC
     - CALL.   PROGRAM CONTINUES.',///)
      GOTO 99
    8 WRITE(6,108)
  108 FORMAT(///,' *****  ATTEMPT TO ASSIGN A NEGATIVE VALUE TO A PARAME
     -TER.   PROGRAM CONTINUES.'///)
   99 RETURN
      END
```

ASSEMBLER SUBROUTINE BAGS

This subroutine provides a mechanism to simulate the random delivery of passenger bags. It is called by each deplaning passenger transaction with one of the argument values specifying the number of bags assigned to the passenger group. BAGS assigns a random integer between 1 and 64 to each simulated bag. The value of the largest random number assigned to the bags of the group is retained by the transaction in PH3. The number of times each integer occurs is recorded in the 64 element array, MH7. When all transactions from a flight have completed calling BAGS, elements of MH7 contain the number of times the corresponding ramdom number between 1 and 64 was generated. The sum of elements of MH7 is identical to the number of bags on the flight. The total number of bags for termination passengers is stored in MH1 (PH2,14) and in MH1 (PH1,15) for transfer passengers. The GPSS main program and FORTM will use the values in the MH7 elements to simulate the time required for bag delivery. The waiting time of the passenger transaction will depend upon the number of simulated bags in each MH7 element, the delivery rate and the highest random number generated for the transaction.

Usage - This subroutine is called by deplaning passenger transactions in the Deplaning Passenger Logic Section of the GPSS main program. A HELP block performs the call as shown in the following example:

HELP BAGS, PH1, FN*PB14, 4, 3, PB8.

The B-operand defines the MH1 row number of the simulated flight deplaned by the transaction. The C-operand specifies the bag distribution function placed in PB14 and passes the value selected by the transaction from the distribution to BAGS. The D- and E-operands specify the number of the byte and the halfword parameter, respectively, to place the number of bags assigned to the transaction and the maximum random number generated by BAGS for the transaction. The F-operand identifies the transaction as representing a terminating or transfer passenger by containing respective values of 1 or 2.

For the above example, subroutine BAGS, returns with the number of bags assigned to PB4 and the maximum random number generated for this transaction in PH3. Elements of MH7 are incremented by each of the transactions associated with the flight if PB4 is non-zero. After all deplaning passenger transactions from the flight have completed the use of BAGS, the flight transaction, operating in the Baggage Unloading Logic Module, executes a HELPA call to FORTM to inspect MH7 and place information about the matrix in byte parameters. The FORTM program resets the MH7 elements to zero values. After the return from FORTM occurs, the flight transaction resets logic switch DPL1C to allow deplaning passenger transactions from the next succeeding flight to execute BAGS.

Restrictions - Subroutine BAGS branches to storage locations internal to IBM GPSS-V. Use of this subroutine with other systems may lead to unpredictable results.

Program Logic -

Subroutine BAGS, after executing the SAVE instruction and declaring register 12 as the base register, tests for the value at LINKADDR for zero to determine if the subroutine has been executed previously. If previously used, the program branches to LINKED. Otherwise it obtains the starting address of MH7 and stores it at the address LINKADDR.

At LINKED, the program loads the value 4096 into register 14 then adds the contents of register 2 to register 14 to satisfy an entry condition for GPSS subroutine DRAND. The B-through F-operand values are loaded into registers 4 through 8, then stored at the 5 fullword locations starting at NORAND. The value of the C-operand is loaded into register 4 and tested for a zero value. If this occurs, no bags are simulated and the program returns to GPSS.

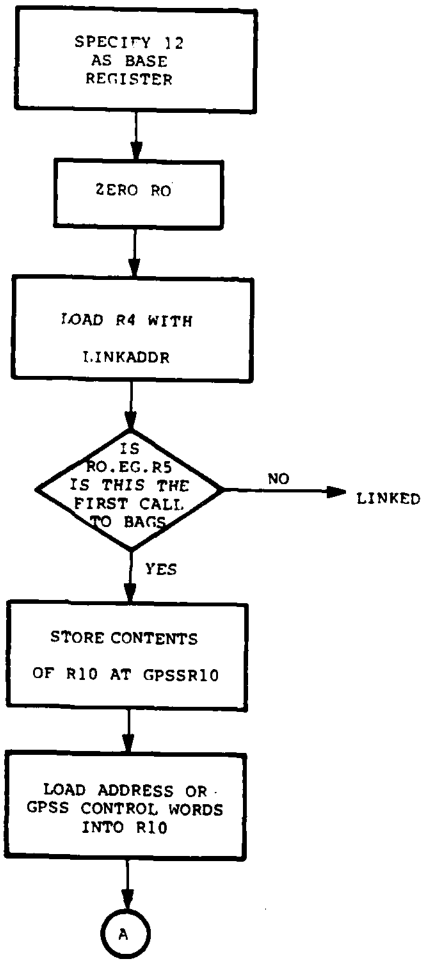
For a non-zero C-operand, the program loads the MH7 starting address into register IO and the PB8 value into register 0, then branches to DRAND at location NEXTRAND to produce a random number using RN8. The random number appears in register 7. The random number is shifted from 0 to 1000 to a range of 0 to 62 by a right shift of four bits.

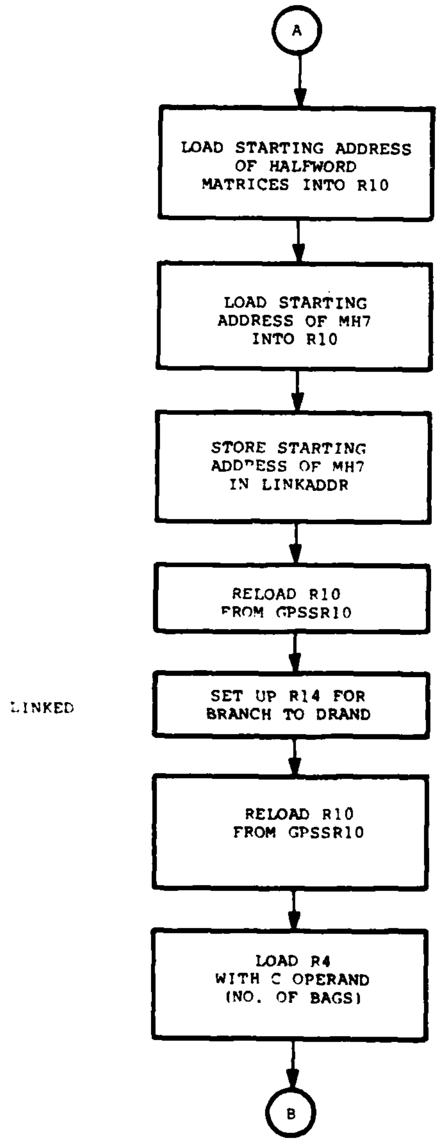
A transfer passenger transaction causes BAGS to branch to XFER. For the terminating passenger case, the program

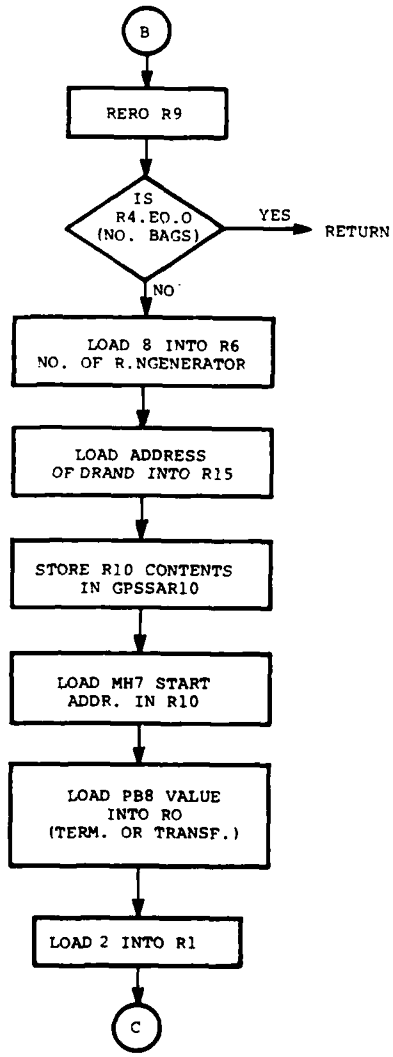
continues and increments the MH7 count of occurrences of the random number. For terminating and transfer passengers the random number selected is compared to contents of register 9 and retained if larger. At location COUNT the program performs a test to determine if an additional random number is to be generated. If true, the program branches back to NEXTTRAND.

The program places the number of bags in PB4 and the highest random number in PB3 by using subroutine STPVAL. It then increments the number of bags in MH1 (NORAND, 14) for terminating passengers or in MH1(NORAND, 15) for transfer passengers. The program then returns to GPSS.

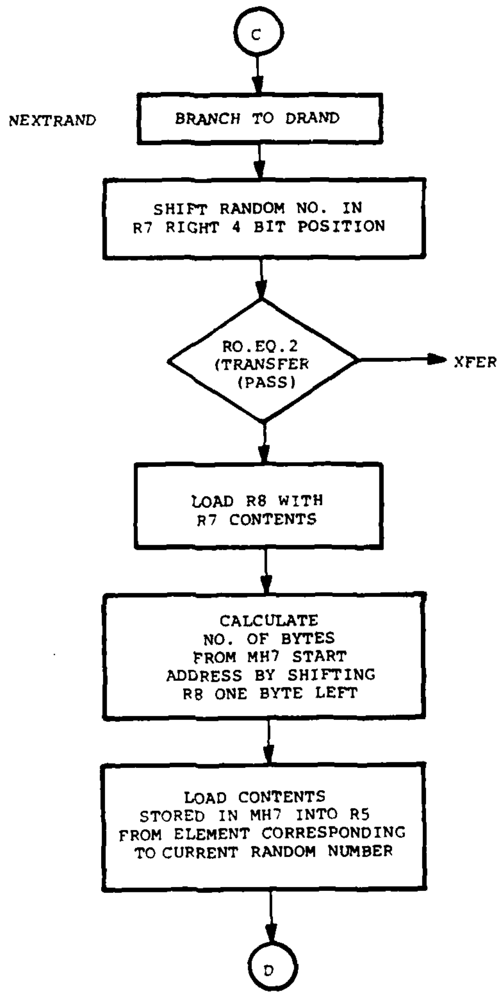
BAGS


SPECIFY 12
AS BASE
REGISTER
ZERO RO
LOAD R4 WITH
LINKADDR
IS
RO.EG.R5
IS THIS THE
FIRST CALL
TO BAGS
NO
LINKED
YES
STORE CONTENTS
OF R10 AT GPSSR10
LOAD ADDRESS OR
GPSS CONTROL WORDS
INTO R10
A

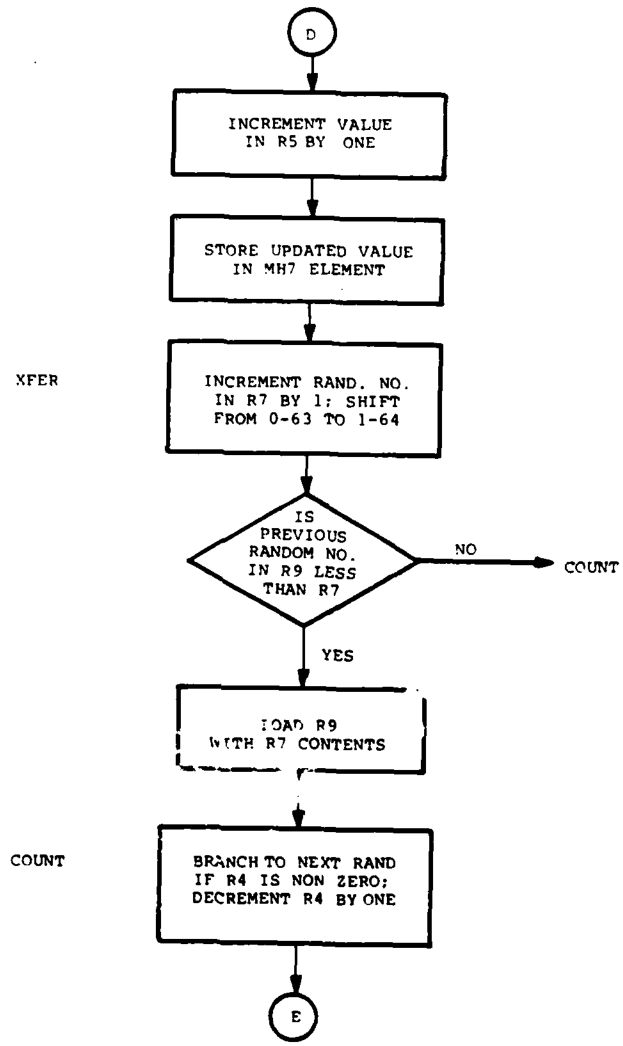

A
LOAD STARTING ADDRESS OF HALFWORD MATRICES INTO R10
LOAD STARTING ADDRESS OF MH7 INTO R10
STORE STARTING ADDRESS OF MH7 IN LINKADDR
RELOAD R10 FROM GPSSR10
LINKED
SET UP R14 FOR BRANCH TO DRAND
RELOAD R10 FROM GPSSR10
LOAD R4 WITH C OPERAND (NO. OF BAGS)
B

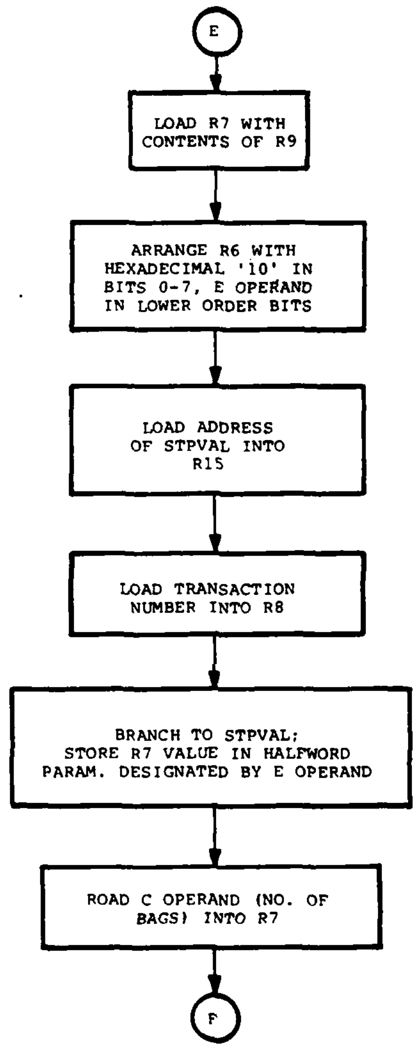


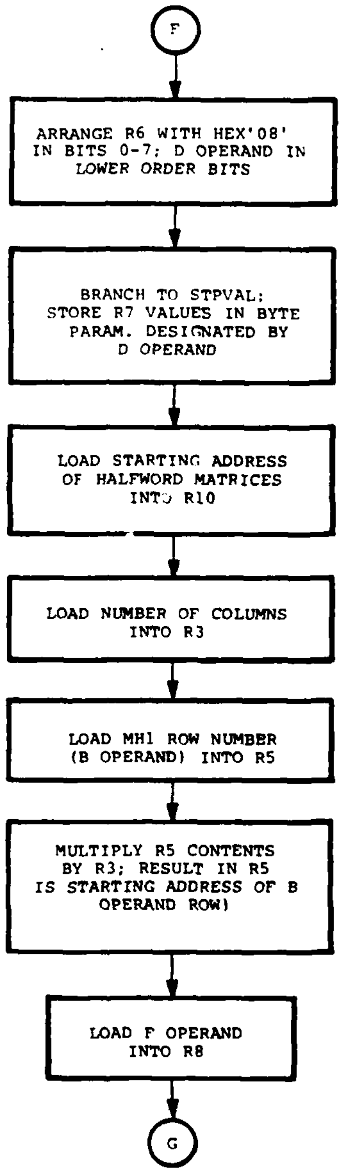
B
RERO R9
IS
R4.EQ.0
(NO. BAGS)
YES
RETURN
NO
LOAD 8 INTO R6
NO. OF R.NGENERATOR
LOAD ADDRESS
OF DRAND INTO R15
STORE R10 CONTENTS
IN GPSSAR10
LOAD MH7 START
ADDR. IN R10
LOAD PB8 VALUE
INTO RO
(TERM. OR TRANSF.)
LOAD 2 INTO R1
C

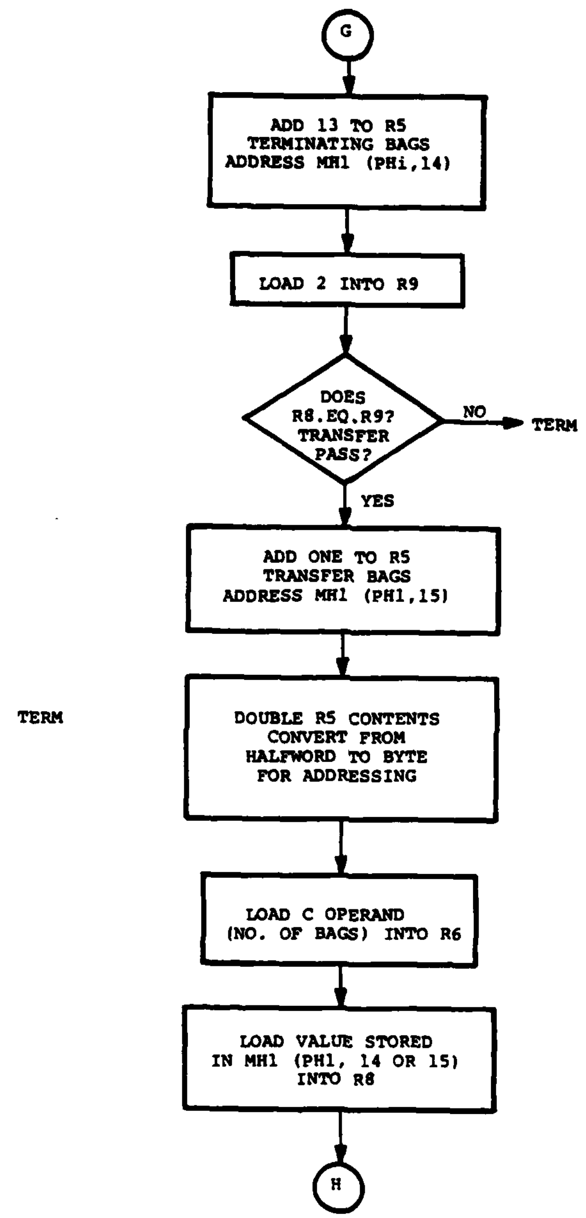


C
NEXTRAND
BRANCH TO DRAND
SHIFT RANDOM NO. IN
R7 RIGHT 4 BIT POSITION
RO.EQ.2
(TRANSFER
(PASS)
XFER
LOAD R8 WITH
R7 CONTENTS
CALCULATE
NO. OF BYTES
FROM MH7 START
ADDRESS BY SHIFTING
R8 ONE BYTE LEFT
LOAD CONTENTS
STORED IN MH7 INTO R5
FROM ELEMENT CORRESPONDING
TO CURRENT RANDOM NUMBER
D

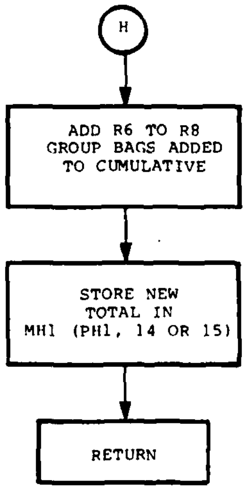

D
INCREMENT VALUE IN R5 BY ONE
STORE UPDATED VALUE IN MH7 ELEMENT
XFER
INCREMENT RAND. NO. IN R7 BY 1; SHIFT FROM 0-63 TO 1-64
IS PREVIOUS RANDOM NO. IN R9 LESS THAN R7
NO
COUNT
YES
LOAD R9 WITH R7 CONTENTS
COUNT
BRANCH TO NEXT RAND IF R4 IS NON ZERO; DECREMENT R4 BY ONE
E


E
LOAD R7 WITH CONTENTS OF R9
ARRANGE R6 WITH HEXADECIMAL '10' IN BITS 0-7, E OPERAND IN LOWER ORDER BITS
LOAD ADDRESS OF STPVAL INTO R15
LOAD TRANSACTION NUMBER INTO R8
BRANCH TO STPVAL; STORE R7 VALUE IN HALFWORD PARAM. DESIGNATED BY E OPERAND
ROAD C OPERAND (NO. OF BAGS) INTO R7
F


F
ARRANGE R6 WITH HEX'08' IN BITS 0-7; D OPERAND IN LOWER ORDER BITS
BRANCH TO STPVAL; STORE R7 VALUES IN BYTE PARAM. DESIGNATED BY D OPERAND
LOAD STARTING ADDRESS OF HALFWORD MATRICES INTO R10
LOAD NUMBER OF COLUMNS INTO R3
LOAD MH1 ROW NUMBER (B OPERAND) INTO R5
MULTIPLY R5 CONTENTS BY R3; RESULT IN R5 IS STARTING ADDRESS OF B OPERAND ROW)
LOAD F OPERAND INTO R8
G



G
ADD 13 TO R5
TERMINATING BAGS
ADDRESS MH1 (PHi,14)
LOAD 2 INTO R9
DOES
R8.EQ.R9?
TRANSFER
PASS?
NO
TERM
YES
ADD ONE TO R5
TRANSFER BAGS
ADDRESS MH1 (PH1,15)
TERM
DOUBLE R5 CONTENTS
CONVERT FROM
HALFWORD TO BYTE
FOR ADDRESSING
LOAD C OPERAND
(NO. OF BAGS) INTO R6
LOAD VALUE STORED
IN MH1 (PH1, 14 OR 15)
INTO R8
H


H
ADD R6 TO R8
GROUP BAGS ADDED
TO CUMULATIVE
STORE NEW
TOTAL IN
MH1 (PH1, 14 OR 15)
RETURN

```
*    CALL:   HELP    BAGS,
*                    FLT NO (MH1 ROW NUMBER),
*                    FN* (NO OF BAGS DISTRIBUTION),
*                    *** PB(NO OF BAGS) *** R E T U R N,
*                    *** PH(MAX RANDOM NO) *** R E T U R N,
*                    PB8 (TERMINATE OR TRANSFER)
*
*            GENERATES NUMBER OF RANDOM NUMBERS SPECIFIED BY B ARG.
*            HELP BLOCK ASSUMES THAT C & D ARGS ARE A PB AND PH NUMBER.
*            RETURNS NUMBER OF RN'S GENERATED IN C ARG, MAX RN IN D ARG.
*            RANDOM NUMBERS ARE IN THE RANGE 1-64.   FOR EACH FLIGHT,
*            FOR TERMINATING PASSENGERS ONLY, C COUNT IS KEPT OF HOW MANY
*            TIMES EACH RANDOM NUMBER IS GENERATED.   THESE COUNTS ARE
*            RETURNED VIA MH7.   THIS INFORMATION IS SUBSEQUENTLY
*            PICKED UP BY THE COPY XAC SPLIT TO BAG CLAIM LOGIC WHICH
*            ALSO RESETS MH7 TO ZEROS.
*            BAGS FOR GIVEN FLT ARE SUMMED IN MH1, ROW FLTNO:
*                                 PB8 EQ 1 ---> COL 14 - TERMINATE BAGS
*                                 PB8 EQ 2 ---> COL 15 - TRANSFER BAGS
*
BAGS     START 0
         SAVE  (14,12)
         BALR  12,0
         USING *,12
         SR    0,0
         L     4,LINKADDR
         CR    0,4
         BNE   LINKED
         ST    10,GPSSR10
         L     10,24(10)
         L     10,1068(10)
         L     10,168(10)        MH 7
         ST    10,LINKADDR
         L     10,GPSSR10
LINKED   L     14,=F'4096'
         AR    14,2
         LM    4,8,0(10)
         STM   4,8,NORAND
         L     4,NORAND+4
         SR    9,9               MAX RANDOM NUMBER
         CR    4,9
         BE    RETURN
         LA    6,8               RN8
         L     15,92(10)
         ST    10,GPSSR10
         L     10,LINKADDR
         L     0,NORAND+16       PB8 VALUE
         LA    1,2               PB8 EQ 2 FOR TRANSFER PAX
NEXTRAND BALR  5,15
         SRA   7,4(0)
         CR    0,1               TEST FOR TRANSFER PAX
         BE    XFER
         LR    8,7
         SLA   8,1(0)
         LH    5,0(8,10)
         A     5,=F'1'
         STH   5,0(8,10)
XFER     A     7,=F'1'           RN 1-64
         CR    9,7
         BNL   COUNT
         LR    9,7               SAVE MAX RN
```

```
COUNT    BCT   4,NEXTRAND
         LR    7,9
         L     9,=X'10000000'   PH
         L     6,NORAND+12
         OR    6,9
         L     10,GPSSR10
         L     15,52(10)
         L     10,24(10)
         LH    8,738(10)       XAC NO
         BALR  5,15
         L     7,NORAND+4
         L     6,NORAND+8
         L     9,=X'08000000'   PB
         OR    6,9
         BALR  5,15
         L     10,1068(10)     MH AREA
         LH    3,30(10)        NO OF COLS IN MH 1
         L     10,24(10)       MH 1 ADDR
         L     5,NORAND        FLIGHT (ROW) NUMBER
         S     5,=F'1'
         MR    4,3             (ROW - 1) * NO OF COLS
         L     8,NORAND+16
         A     5,=F'13'        MH1(*,14) FOR TERM PAX BAGS; ADD COL - 1
         L     9,=F'2'         PB8=2 ---> TRANSFER PAX
         CR    8,9
         BNE   TERM
         A     5,=F'1'         MH1(*,15) FOR TRANSFER PAX BAGS
TERM     AR    5,5
         L     6,NORAND+4
         LH    8,0(10,5)
         AR    6,8
         STH   6,0(10,5)
RETURN   RETURN (14,12)
NORAND   DS    5F
GPSSR10  DS    1F
LINKADDR DC    X'00000000'
         END
```

DATE

FILMED

8